THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 6, July–August 1983
Printed in U.S.A.

Human Factors and Behavioral Science:

Statistical Semantics: Analysis of the Potential Performance of Key-Word Information Systems

By G. W. FURNAS,* T. K. LANDAUER,* L. M. GOMEZ,* and S. T. DUMAIS*

(Manuscript received February 10, 1982)

This paper examines how imprecision in the way humans name things might limit how well a computer can guess to what they are referring. People were asked to name things in a variety of domains: instructions for text-editing operations, index words for cooking recipes, categories for "want ads," and descriptions of common objects. We found that random pairs of people used the same word for an object only 10 to 20 percent of the time. But we also found that hit rates could be increased threefold by using norms on naming to pick optimal names, by recognizing as many of the users' various words as possible, and by allowing the user and the system several guesses in trying to hit upon the desired target.

I. INTRODUCTION

Computer-based information management systems can store, manipulate, and transmit enormous quantities of information. They can allow almost unlimited organization, multiple indexing and cross referencing, and are capable of performing rapid and complex search operations. Thus, they can provide far more powerful tools for knowl-

* Bell Laboratories.

edge management than have been available previously. Such tools will be important to almost everybody: cooks wanting to find recipes, doctors needing patients' histories, managers tracking inventories, clerks filling orders and keeping records, and buyers looking for products. It would be fine if such systems could be used directly by inexperienced and occasional users, people whose main job or talents lie elsewhere. The hope is that new information-seeking tools could be operated with no more training than is needed to use a card-file or a book index, but with much greater success.

We believe that some of the greatest difficulties blocking this dream are psychological. Certainly, more available speed and capacity, and new algorithmic and data structure developments involving deep and difficult problems will be needed before the arrival of fully satisfying information systems. But, progress along these lines has already been enormous and continues at a brisk pace. Meanwhile, although even existing computational capabilities are very great, our ability to make them easily usable by nonspecialists has been quite limited. An important psychological problem is in understanding the relationship between what people say and what they want. This understanding is the key to designing systems that can infer what services or information users need from the input they provide. This basic capability is needed by information systems of many sorts, bibliographic reference search systems, business management databases, airline reservation systems, plant inventory and customer record systems, and even text processors. We believe that current systems generally fall very far short of the ideal of always knowing just what to give the user.

At this time although it is clear that there is a problem, very little relevant work has been done (see Carroll[1] and Landauer, Galotti, and Hartwell[2]). We do not even know the locus of the problem in the chain of actors and events. For example, the main problems may lie at the source; perhaps the human intellect is basically incapable of forming information specifications that are very precise. If so, perhaps no system could do much better than the current ones. But, it is also possible to believe that if systems could understand people as well as, say, close friends understand each other, there would be much less of a problem.

We have tried to get some idea of the extent of this problem by attacking a small but important part of it. We have asked people to give descriptions of various information objects, and analyzed their responses to determine how well the objects to which they refer can be inferred from what they say. We have begun by studying the referential properties of isolated words and short phrases.

Our goals in this research have been twofold; first, to advance understanding of the psychological processes by which human seman-

tic reference is generated, and second, to model and estimate the strengths and weaknesses of information systems that take human-generated descriptions of sought items as their input. Empirical observations of naming behavior provide the necessary data for both enterprises.

In this article we first describe four sets of object-description data, the way they were collected and reduced, and some of their more interesting features. Then we present a series of analyses in which we treat the collected descriptions both as representative of what users would provide as input to information retrieval systems, and as the source of the information that the system would use in determining user wants. The hypothetical systems we consider are limited to ones in which the user's initial entry or query consists of a single word or short phrase. We do not attempt to analyze the possible performance of systems that make use of sentential syntax, or linguistic or real world context. The reason for this limitation is largely pragmatic; it postpones analysis of many difficult complexities. However, the characteristics and limitations of single-word to object reference that we have investigated have strong implications for many access methods (by which we mean the access method provided for the user, not that used by the program). We are, of course, aware that there are data access methods that do not start with the user entering a key word or phrase; for example, there are strictly menu-driven systems, ones that rely on well-formed queries and restricted query languages, and also ones that attempt some form of natural language understanding. These other methods share some, but not all, of the same conceptual and practical difficulties of key-word methods, and each raises somewhat unique and interesting psychological issues of its own. Where our data bear on these issues in reasonably direct ways, we offer some comments, but we focus primarily on the key-word comprehension process and its ramifications. In the final section of this paper, we discuss some of the reasons why key-word access is as limited as we find it to be and consider several potential methods for overcoming the deficiencies.

II. DESCRIPTION OF DATA SETS

There are things a computer system has or does to which a user might wish to refer. These "information objects" are just the objects, e.g., operations or data sets, to which commands and queries apply. We have collected descriptions of information objects in four quite disparate domains, chosen in an intentional effort to achieve variety, and also for their relevance to a number of special problems that are outside the concerns of this paper. The four sets are: the verbs used in spontaneous descriptions of the operations needed to perform

manual text-editing operations, descriptions of named common objects designed to induce another person or a computer to return the name in a game like the *PASSWORD*™ game, superordinate category names for items available in a swap-and-sale listing similar to classified ads in newspapers, and index words provided for a set of main-course cooking recipes.

In this section we will describe how the object specifications were obtained from people, how they were reduced to single-word or short-phrase keys, and then summarize a few qualitative and statistical characteristics of the responses.

2.1 Text-editing operations

The first data came from language applied to text editing. The study was one of two conducted to explore "naturalness" in command names.[2] Forty-eight secretarial and high school students were asked to provide instructions to another hypothetical typist describing what operations needed to be performed on text marked by an author for correction. These corrections involved two examples each of 25 sorts of edits: five basic operations (insert, delete, move, change, and transpose) on each of five textual units (blanks, characters, words, lines, and paragraphs).

Preprocessing in this case reduced each response to the main verb or phrase in the instruction describing how to perform the editing operation. While this was in fact accomplished manually here, we believe that a simple parser and English word list could in principle have given nearly identical results. These expressions may be considered candidates for command names for editing systems.

Perhaps the most striking result was that there was extensive disagreement in the verbs people produced. This point is the main focus of the current article, and will be dealt with in considerable detail later. For now let us just make a few preliminary notes, e.g., that the three most popular names for each operation accounted for only 33 percent of the total number of responses. The intersubject agreement, the probability that any two people used the same verb in describing a particular text correction, is only .08. Since each of the 25 sorts of edits occurred twice, we also have a measure of within-subject (with 1200 observations) agreement. The probability that an individual subject used the same main verb in the two cases was .34.

What agreement there was did not favor the terminology used by our locally popular editor (the *UNIX** Operating System text editor ed). For 24 out of 25 of the types of edits, the name in ed was not the most frequent spontaneously given name. Use of the terms "delete" and "substitute" was quite rare, for example. (Landauer et al.[2] went

* Trademark of Bell Laboratories.

on to show, however, that this caused no problems initially in learning the basic editor.) People preferred "add" for the insert operations, "omit" for delete operations, and "change" for the replace operations. There was little consensus in describing the transpose and move operations.

2.2 Common object descriptions

These data were originally collected in a study by Dumais and Landauer[3] that examined how people naturally obtain information from one another. The information objects here were names of 50 common items chosen from 10 "categories." The categories were: cities, proper names, clothing, animals, food, household items, abstract words, a category of words with highly associated opposites (e.g., black, love), and two categories whose members were words selected in such a way that negation might figure strongly in the descriptions, e.g., to eliminate the unwanted set members. A total of 337 New York University students were asked to write down a description that would enable another person (or in half the cases a hypothetical computer) to guess the object. There were no restrictions as to the form or content of the descriptions, except that they could not contain the target word itself. Subjects also indicated whether they had any computer experience. Those with at least one computer course were classified as computer-"experienced" in the data summaries discussed below.

A subsequent study was conducted to evaluate the effectiveness of the descriptions generated in the first study. Twenty-five subjects (6 Murray Hill area homemakers and 19 employees of Bell Laboratories) were each given 150 descriptions randomly selected from those generated by the NYU students. They were asked to: (1) guess (without knowing the alternatives) the item being described, and (2) indicate on a five-point scale their confidence in their guess.

Principal results from the main study include the finding that when communications were intended for computers, people with computer experience were relatively more terse, and nonexperienced people were relatively more verbose than when communications were intended for people. However, there was no simple relationship between verbosity and effectiveness, i.e., guessing accuracy as indicated by the second study. People were somewhat more successful in guessing the target items when the descriptions had been provided by people without computer experience (81.2 percent vs. 78.5 percent), but this difference is not statistically significant.

A point of considerable interest here was the style of specification that subjects used. Subjects' descriptions were not very precise; typically they refer to a whole set of items, not just the intended target (although we have no good measure of this other than informal ratings

of denotative class size). Still, the average successful guess rate of the second group of subjects was over 80 percent. We will return to a discussion of this paradoxically high success rate towards the end of this paper. The most frequent way of specifying target items, used about 60 percent of the time, was to describe them in terms of a superordinate (sometimes followed by characteristics or attributes that distinguish the intended target from other members of the superordinate category). Another fairly common form of description (~20 percent) was the use of exemplars. For several of the target words (e.g., motorcycle, magazine, sports, games, science), subjects listed examples of more specific items falling into the target category (e.g., Harley, Suzuki ..., in the case of motorcycle) instead of attempting a more formal definition. Negations (and opposites) were used less than 50 percent of the time for the words we thought were particularly amenable to this form of description.

For the purposes of the analyses undertaken in the current paper, these descriptions were preprocessed to merge minor variations in as automatic a fashion as possible: uppercase was folded to lowercase, word endings were stripped (plurals, tense markers, etc.), and "non-content" words (including articles, imperatives, conjunctions, prepositions, pronouns, and tenses of the verb "to be") were removed.

An average of 8 words per description were in this way condensed to an average of 5.4 words. The first of the remaining words (i.e., first standardized content word) was tabulated for the statistical analyses.

2.3 Superordinate categories for swap-and-sale items

The major purpose of collecting these data was to develop empirical networks of "ISA" relations, that is, classification hierarchies based on user knowledge and representations, for a set of items to be incorporated in an experimental menu-driven information access system.[4]

The information objects were 64 items taken randomly from roughly 300 entries on a monthly bulletin board listing of items for swap and sale at Bell Laboratories. The subjects were 30 local New Jersey homemakers. Each subject worked with a random 32 of the 64 target items. They were told that they were participating in the study to find out how they classified various items being sold on local bulletin boards. The use of these categories in helping people in future computerized retrieval systems was mentioned. Subjects were instructed to complete successive "All ____ are ____" sentences. Beginning with the specific target they were to give its immediate superordinate (e.g., "all red Delicious apples for sale @10¢ ea. are apples"). Then they copied the first given superordinate ("apples") to the beginning of the next incomplete sentence and finished the new sentence with a still

more general category (e.g., "all apples are fruit"). They were to continue in this way (e.g., "all fruits are food") until they could go no further. They were then to go back and find some category that had another superordinate in addition to the one they had already cited, and list that category with its new superordinate. These data were also standardized by stripping off endings and discarding noncontent words.

On average, an individual subject produced 2.1 different chains of successively more general superordinate categories for each stimulus. The chains averaged 2.5 superordinates each, with superordinate categories named in phrases containing an average of 1.7 standardized (content) words. For the current paper, only the lowest level (most specific) category, from the first generated chain of superordinates, was used. The category name was used in its (standardized) entirety.

The categories given in this study make it apparent that the construction of a network that will faithfully match all users' conceptions of a domain is not an easy matter. People have difficulty in generating superordinates and show considerable disagreement as to how things should be grouped under those superordinates. Categorization and indexing schemes currently in use always depend on a user's either generating the same superordinate as the system knows about, or at least being able to choose the right one from a list. Perhaps the difficulty and lack of agreement among people in categorizing information objects account for much of the perceived deficiency of current menu-driven data access methods.

2.4 Recipe index words

The original motive for this data set was to study the effect of domain expertise (i.e., cooking skill) on indexing and key-word usage.[5] The information objects were 188 main-course cooking recipes (French, Italian, Mexican, and American cuisine) taken from 12 cookbooks of explicitly varied sophistication (ranging from a garden club's cookbook,[6] through *The New York Times Cookbook*,[7] to *The Art of French Cooking*[8]).

There were three groups of eight subjects each: experts, who taught cooking classes; and intermediates and novices, selected from local homemakers who came out at the high and low extremes of several self-rating scales on culinary sophistication.

Subjects were told their task was to describe each of the recipes in key-word form, selecting at least three but no more than seven descriptive words or brief phrases for each recipe. They were told that their job was similar to that of a librarian who is creating an index or card catalog, and that the descriptions should be useful to another person trying to locate that recipe in a large set of recipes. Half of the subjects

in each experience group were instructed to direct their descriptions to expert cooks using the index, the other half to novice cooks. The description task was self-paced by each subject in her home. Subjects required between 5 and 10 hours to complete the task.

Terms here were again preprocessed to remove word endings and noncontent words. All multiple-word productions were scored by two judges (the experimenters) to determine if the phrase could be decomposed into its constituent words and maintain its meaning. Subjects produced an average of 5.4 key words per recipe, the first, "most important" of which was studied here.

Again we found considerable diversity. For the 188 recipes, a total of 303 different word types were used by the 8 experts, 220 by the 8 intermediates, and 252 by the 8 novices. It is interesting to speculate on the reasons why these groups differ. Perhaps the experts have a large and specialized vocabulary and the novices have an unruly, haphazard one. In any case, there seems to be something more conventional about the word use of intermediates, a point to which we will return later.

2.5 General comments on the data sets

These data sets all pertain to information objects that one might want accessible on a computer. They were also all of modest size. Other than that, though, they tapped very different knowledge domains, they asked for specification in a number of different ways, and they were provided by different kinds of people. Moreover, the method of reducing the free-form descriptions given by our participants to single words and short phrases varied somewhat from one case to the other. This variety of data is important for our purposes. In order for results to have any pretense of robustness, it is important that they be obtained on a sufficient variety of cases to assure that it is not the particulars of the objects at hand that are responsible for the observed characteristics. We know of no way to actually sample data domains, descriptive methods, and reduction methods in a representative way. However, we believe that results that hold for all of the disparate sets that we have studied stand a good chance of holding for most others.

For each domain, our data can be represented as a table in which the rows are words provided by the subject, the columns are the objects to which these words were applied as descriptions, and the cell entries indicate the number of times each word was used in the description of a given information object. The questions we ask concern how the information contained in such a table might be used to guess from an input word what object is intended. Two partial tables are shown in Tables Ia and b. Table Ia is derived from the text-editing study (for five objects) and Table Ib from the common object data. The numbers

Table I—Word-object data

(a) Sample data from the text-editing study

Words	Objects				
	insert	delete	replace	move	transpose
change	30	22	60	30	41
remove	0	21	12	17	5
spell	4	14	13	12	10
reverse	0	0	0	0	27
leave	10	0	0	1	0
make into	0	4	0	0	1
...					

(b) Sample data from the common object study

Words	Objects					
	calculator	lime	Lucille Ball	pear	raisin	robin...
small	17	0	0	0	7	4
machine	4	0	0	0	0	0
green	0	18	0	7	0	0
bird	0	0	0	0	0	21
fruit	0	1	0	19	1	0
red	0	0	8	0	0	7
female	0	0	2	0	0	0
...						

in the tables represent the frequency with which a word was used to refer to an object.

In fact, there is an implicit third dimension to these tables, representing the person from whom the description was obtained, and sometimes a fourth dimension, representing which of several words of a multiple-word description provided by a given subject is involved. However, for most of the analyses we consider only the first word given, and the matrices are all very sparse, so we have chosen to collapse across subjects. It is worth noting that the tables are not sparse simply because we have failed to collect enough data. Word usage tends to resemble Zipf's distribution[9] (supposedly straight line relation between occurrence frequency and rank frequency when both are plotted logarithmically) in that a few words are used very frequently and many words (over 340 here) used only once (see Fig. 1). As more and more data are collected some cells increase in frequency, but the number of unique words also grows so that the sparseness of the table tends to be preserved. Moreover, most words refer only to a limited number of objects, so that such tables usually have a large number of empty cells.

III. INTRODUCTION TO ANALYSES TO DETERMINE REFERENT EFFECTIVENESS

In the analyses that we report in Section IV, we have been interested in how much information about referent object identity is contained

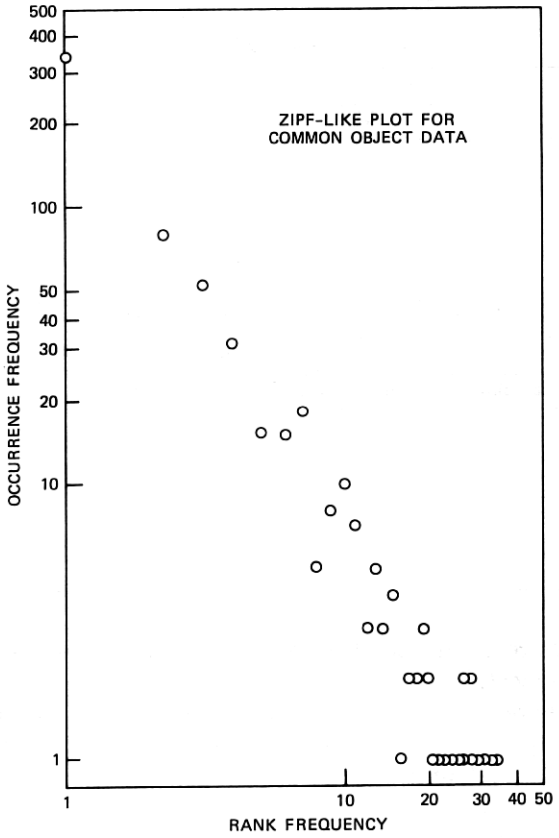

Fig. 1—Plot of common object word usage data.

in the words used to describe them. To consider how good a word or expression is as a reference to some object, it is necessary to suppose some sort of a mechanism by which the expression is comprehended. We can only estimate the value of inputs as determinants of output if we can specify the function that takes one into the other. The inputs of our problem are the words provided by people in specifying information objects. The output is a guess or set of guesses about which object was intended. Our approach has been to specify various ways in which the input can be made to yield the output. We have considered models or functions that are intended to mimic what happens in typical information systems and also ones that try to improve on current methods in various ways. We also develop models that allow us to estimate what ideal input/output mappings could accomplish. It is in this sense of ideal or asymptotic performance that we can appraise the limits of word reference.

The models that we examine have in common that they use the frequency table of users' word reference, and only this data, to guide the system in guessing users' intents.

3.1 The general model

In the sections below we consider a number of special cases of the following rather general baseline model. To retrieve a desired item the user is assumed to enter one (though in some instances, K different) key word(s). The system is assumed to make either one or M guesses as to the user's intent. Success is counted if the intended object is among the system guesses.

As an actual system, our baseline model would operate as follows. The user would enter one (or more) key words or phrases. If it could recognize the input at all, the system would return one or more potential objects according to some rule, perhaps relying on data in a table of observed word-use frequencies. (If it could not recognize the input, it would return nothing.) In cases where the system returns several alternative objects, it can be assumed that the user would be able to select from among them the one intended without error. This corresponds to a simple case of what might be called a "menu-on-the-fly" technique, in which the first entry is a freely chosen key word, and the next step is a choice among a menu of items selected on the basis of the system's knowledge about referents the human user might have intended by that key word.

We will now describe the models that we have analyzed, beginning with the simplest ones, those closest to the way typical systems are currently set up, and proceed to more sophisticated models. Each model is characterized by a set of assumptions or constraints on the input/output function of the system.

Sections 3.2 to 4.7 consider a number of models and analyses in considerable technical detail. Some readers may wish to skip these sections and continue to a summary of the highlights in Section 4.8.

To give a preview, the results convince us of three major points. First, the vocabulary problem is a serious one for untutored users of computers, particularly given the poor naming techniques typical of current practice. Second, the difficulties come from a severe and fundamental lack of consensus in the language community on what to call things. Finally, our research suggests that, by using a very non-standard approach, namely trying to make the best possible guesses on all user words, substantial improvements could be made in meeting the underlying objective of providing access to the things people want.

3.2 Formalizing the general model

Let us make things more formal. We have two sorts of entities, the user-generated inputs, referred to here as "words," "terms," or "de-

scriptors," and the system outputs, called "information objects," "targets," or just "objects." We are concerned with two relationships between these entities. On the system's side are the associations dictating how the system will respond to a user's words—which words will be associated with what system objects. On the user's side are the actual intended associations—what the user in fact meant by a given word. Both the system's and the user's relationships can be shown in the form of a table. The *system table* gives the input/output mapping of which object the system will retrieve (execute, etc.) for what words, with a 1 wherever a word is associated with a system object, and a 0 otherwise. The *user table* shows the frequencies with which people say what words for what objects. Examples of a user table were shown in Table I. This table can be used in the evaluation, and, if one wishes, in the design of the system table.

3.2.1 The system table

The system table has two aspects. First is what might be called the set of structural constraints. These concern how much and what variety of input the system will respond to, and are imposed by capacity limitations on memory, processing, data collection, or sometimes by the computer algorithms used. Second is the choice of a particular instance of the input/output system table, that is, the assignment of exactly which words the system will respond to, and with what objects. One design problem centers on whether assignments will be made in an effective or possibly optimal way.

Structural constraints in the system's input/output table correspond to general restrictions on which cells may be used in the mapping of input to output. We consider patterns based on limiting the number of cells in rows and columns (e.g., the number of words understood as referring to each object and the number of objects taken as possible referents for each word). In particular, we will be considering models where, in the system table, there are various restrictions on:

1. Total words recognized by the system (number of nonzero rows in the system table)
2. Number of words recognized for each object (column totals)
3. Total objects referenced (number of nonzero columns)
4. Number of objects referenced by each word (row totals)
5. Total word-object associations "understood" by the system (grand total of cells).

Once a general structure has been determined, one still must choose which associations to give the system, i.e., exactly which cells to include in the mapping. We consider three cases that give rise to different versions of each model. The first is a purely random assignment. Within the specified structural constraints, the cells defining

the system's associations between words and objects are chosen randomly. Clearly, this is not a realistic system model, but it is useful in the conceptual analysis of other models and as a baseline of comparison. The second case is a weighted random assignment, assigning a word to an object with the same probability that the word and object occur together in usage. That is, input/output associations are chosen with probabilities proportional to their frequency of use, as indicated by the user table. This is, in fact, a very important case to consider, since it approximates the way many systems are currently designed. We will often refer to this as the "armchair" method: a single human designer sits in his or her proverbial armchair and makes the name-object associations by some sort of intuitive guessing. The user table is in effect a compendium of many humans' "armchair" attempts to assign good words to the system objects. A weighted random sample from the table thus corresponds to a random person's "armchair" nomination. Such weighted sampling therefore allows us to estimate the effectiveness of "armchair" design. The last assignment technique is by the best available optimization procedure. Optimization methods make generous use of the empirical data in the user table to pick the best cells. For some system constraints we do not know a combinatorially feasible way to find the best configuration, but it is possible to improve substantially upon the weighted random method. Note that the success of any optimization attempt is also limited by the quality of the data available in the user table.

3.2.2 The user table

The user table is a data matrix compiled from the studies mentioned earlier in this paper. It records the frequency with which untrained users employed various terms in referring to the system objects.

We assume that the descriptions given by our subjects for the information objects in the various sets bear a close resemblance to the entries similar users would make in trying to specify the same objects in a database system. This assumption may be wrong in detail; in using actual systems people might give descriptions somewhat different from those induced by our instructions. However, when we varied the intended recipient of the descriptions, as in the recipe (experts, novices) and common object data (people, machines), there were only small changes in the descriptions. So we believe that descriptive language would change little in attempts to communicate with real, as compared to hypothetical, systems. (We cannot, however, estimate the effect that prolonged interaction with a given system might have on a user's vocabulary.)

We also assume that collection of user descriptions is a good basis for predicting actual user intent from key-word input. In approaching

a data access system, the user must have some information object in mind, and must give a description to the system. In actual use, it is difficult to know what the user really has in mind. Sometimes the user has no clear idea at all. It is not apparent whether this "diffuse target" case is easier or harder for a system to handle. A user with an unclear goal may or may not be more easily satisfied, but is probably less likely to give precise specifications.

It is obvious, however, that the diffuse target case is harder to study. Reliably inducing such a state in the user is difficult, and the system's success is not easily appraised. Therefore, we have collected data and investigated models based on the clear target case.

Thus, in these models it seems to us that the best information the system could start with is to know what typical users would give as descriptions, given the intended object was well known and clearly defined. Our assumption is that the descriptions people give of objects when we specify them will closely resemble the description they would give if they had thought of the specific objects themselves. This seemed a practicable and, we hope, realistic starting point.

3.2.3 Going through the models

Insofar as possible, we will use a consistent expository framework to describe each model. The framework is as follows:

1. Name: A mnemonic title, by which the model will be referenced.
2. Interpretation: What the model is all about.
3. Motivations: Where and why it might come up.
4. Structural constraints: The general form of the system's input/output table ("system table") for the model. This will be given as clearly as possible in English, with a more formal summary of the constraints for each model appearing in Appendix A. This appendix is a very useful reference source in comparing the models.
5. Analyses: For each of three methods of selecting cells for the system table (random, weighted random, and optimized), the following information will be given whenever possible.
   (a) Version: what the method means in this model.
   (b) Evaluation by: what statistic is used to assess its success.
   (c) Result: Statistics are computed for the four data sets:
      (1) Edit command data
         (a) All 25 Operation x Text-Unit combinations (abbreviated: "Ed25")
         (b) For purposes of comparison we include a second analysis of the same data: the five basic operations collapsed across textual units ("Ed5")
      (2) Common object data, all 50 objects ("CmOb")
      (3) Swap-and-sale data, the 64 sale items ("Swap")
      (4) Recipe data, the 188 main course recipes ("Recp")

The following statistics will be given:

Recall probability: how often the method succeeds in returning the desired object.

Mean number returned: when the system is able to make a guess about the user's desired object; this is a record of how many guesses it makes to achieve the reported probability of recall.

A few comments are needed about the "number returned" statistic. First note its importance in considering recall success probabilities. If the system returns a large number of guesses, it can obviously be expected to have a greater chance of including the target. Thus, comparisons of performance demand that this number be kept in mind. Second, note that these indicate the number of things returned when the system in fact recognizes the user's input and so is in fact able to hazard a guess. For some systems it will be common to have insufficient data and to make no guess at all for many user words.

We denote the number of objects, or columns, in the table by C, and the number of user words, or rows, by R. The former is in part well defined by the designers of the system but the latter, the vocabulary size, is usually an artifact of data collection, as it would tend to expand if more data were collected. Here it refers to the size of the vocabulary in the corpus of data we collected.

IV. ANALYSES TO DETERMINE REFERENT EFFECTIVENESS

4.1 Model 1: "One name per object."

4.1.1 Interpretation

Each object in the system is assigned a single term or name. The user enters one term. Success depends on the user's word coinciding with the system's.

4.1.2 Motivations

This approach is common in computer systems—each entity has one and only one name. The name is usually chosen by the designer or by an expert indexer. The designer hopes to establish a convention about what system objects will be called. Users must either learn or guess the names to make the system work. Such learning may be feasible in small systems or for highly practiced users. The growing community of novice and infrequent users, however, are often reduced to trying to second-guess the system, even to find documentation. This is often frustrating and we shall see that in principle the approach is far from adequate. Moreover, the real difficulties of such a scheme often go unremarked simply because traditional computer users have come to accept as normal the necessity of extended learning, repeated second-guessing, lengthy searches, or expert consultation in finding the correct names for programming commands, file names, or information categories.

4.1.3 Structure

The only constraint is that there be exactly one word for each object. Note especially that this model allows the possibility that any given word can be used to name more than one object. This is a situation that designers often try to avoid. It nevertheless arises in several situations, as when programs, commands, file names, or index categories are assigned by many independent users, or to highly similar objects (like bibliographic subject or author specifications, or the case of several functions being collapsed under a single command name). We evaluate models that disallow such "collisions" later. Note that for the likelihood of recalling a given object at all, the aspect of the problem on which we focus in these first analyses, assigning a word to more than one object as we do here, can only improve the expected system performance.

4.1.4 Analyses

4.1.4.1 Version: random. For each object one name is chosen randomly from the total vocabulary for that object, i.e., one cell from each column. That is, the name of each object is a random choice among all the total set of descriptors given to all the objects.

Evaluation: by theoretical value. In this case we can calculate the expected performance of the system exactly. The success rate is given simply by the ratio of t, the total number of cells included in the system mapping, to RC, the total number of cells in the matrix. A simple proof of this appears in Appendix B. See Results Table 1.

Results Table 1

recall probability =	$C/RC = 1/R$, i.e., the reciprocal of the total number of words that users use for all the objects
	Ed5 Ed25 CmOb Swap Recp
recall probability =	.012 .008 .002 .001 .001
mean number returned =	$1 + (C - 1)/R$

The "mean number returned" is a measure of the amount of ambiguity or imprecision in the terms as the system understands them. It is the average number of things the system knows by a given name that the system recognizes; the system must return all of these objects when trying to guess a target. For some of our models the number of objects that the system returns will be fixed ahead of time by design, but here it is the mean of a random variable, easily calculated to be $1 + (C - 1)/R$.*

* Under the pure random method, each object has one of the R words associated with it randomly and independently. Thus, for each object, any given word arising or not becomes a Bernoulli event with probability $1/R$. So having chosen a word for one object,

4.1.4.2 Version: weighted random. A system name is assigned to an object with the same probability that users attributed the term to the object. Here the user enters one key word; and the system has had a key word assigned to each object, based on one other person's nomination. This model mimics, more or less, a currently fairly typical situation for key-word systems (or program-name or command-name access systems) in which the system designer has provided an entry name for each object, obtained only from one usage datum (the designer or indexer's "armchair" introspection), and the user is required to enter just that name. We assume that system designers or indexers are like our subjects in their choice of names, so that the relative frequency with which a name was given to an object by our subjects provides an estimate of the likelihood that a designer would choose that name for that object. (One source of support for this assumption is that experts gave no more consistent names than novices; see below.)

Evaluation: by the "column repeat rate" statistic. We estimated how well such a system is likely to work by estimating the probability that a given word chosen randomly from our population (e.g., by one user on one occasion) would match another word chosen from the same population (e.g., by one designer). This probability is known as the repeat rate.[10] If we index rows (words) by i, and columns (objects) by j, in a word-by-object table, then repeat rate for a given object, rep_j is defined as:

$$r_j = \sum_i p_{ij}^2.$$

This formula can be understood by considering that a match between two randomly chosen words occurs when, for any given word first chosen, the second word is the same. Say the first word was $word_i$; the second word will match it with the probability of $word_i$ occurring in the population, p_{ij}. The probability of a $word_i$ being the first word is also p_{ij}, so the probability of $word_i$ being involved in a match is p_{ij}^2. Summing across all possible words, we get the equation given above.

An unbiased estimate of the population probability of such a match comes from calculating the true probability of such a coincidence in drawing from our sample without replacement, given by:

---

the number of the other $C - 1$ objects having randomly been assigned that same word is given by a binomial distribution with parameters $p = 1/R$ and $C - 1$. Thus the mean number of other objects with the same name as our given object is the mean of this distribution, $(C - 1)/R$. The system will return any of these objects plus the original object, or $1 + (C - 1)/R$ objects.

$$\hat{r}_j = \sum_i \frac{n_{ij}}{N} \frac{n_{ij} - 1}{N - 1},$$

where N is a column total and n_{ij} is the frequency of the ith cell in the column j.

Here we are interested in the average probability of success, given any particular target, throughout the table. So we decompose the overall probability by conditionalizing on columns, calculating the repeat rate for each column, and then average these success probabilities using weights proportional to the column total frequencies. (In our data these column totals are approximately equal by design.) These weighted average column repeat rates are given below for our four sets of data. These numbers may be interpreted as answering the following question: Given that all objects are equally often the desired target, what is the probability that the name given by a user trying to specify a target would match the name assigned to it by a designer? See Results Table 2.

Results Table 2

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.07	.11	.12	.14	.18

While there is some variation among the values, they are all quite small. People do not agree with one another very well as to the first word or phrase with which to label an object. The probability that two typists will use the same main verb in describing an edit operation is less than one in fifteen. The probability that two people will use the same first key word for a recipe is less than one in five. (These numbers also tell us something about the size of the set of alternatives that people use in their disagreement. It is a property of repeat rate that the set of alternatives must be at least $1/r$, and can be quite a bit larger if they are not equally likely, as is the case here.)

Most of the interesting comparisons will be between the different models presented (e.g., this model with the subsequent ones), and not the different data sets. To facilitate model-to-model comparison, all results appear together in a summary table in Appendix C.* The reader is strongly encouraged to refer to this table throughout Section IV.

* Of course, some comparisons between data sets are also of interest. Note for example that the value for Ed25 is higher than for Ed5. This says that the set of words applied to the individual objects is more sharply restricted than for the collapsed classes. This is to be expected, since any diversity between objects in the pattern of terms applied becomes within-class diversity, when the objects are collapsed together, driving the column repeat rate down. Only if all objects in a class had identical naming patterns would the repeat rate not decrease.

The mimicked method is one in which the designer provides the system with only one entry word that it can understand, and the user enters just one key word. This is clearly unsatisfactory for untrained users. The usual solution has been for system designers to rely on users learning the chosen vocabulary, i.e., to try to force the user's table to adapt to a fairly arbitrary system table. When the system is small and the user's interaction frequent, this can work quite well. Indeed, Landauer et al.[2] have shown that using unrelated random names has little or no detrimental effect on initially learning to operate a small editor. But, if the system is large and its use intermittent and nonexpert (as for example in large-scale information retrieval systems like library catalogs, recipe files, or classified product catalogs), it is simply unreasonable to require users to learn a specialized vocabulary.* Despite the designers intentions, the uninitiated will try to make the system work without memorizing extensive naming conventions. Thus the problem remains a real one.

One approach we might consider at this point is to seek expert advice in choosing names. This is a fairly common approach, taken in the hope that experts in a given subject area know what things are, or should be, called and so might generate words of more general currency. Indeed, the indexes to books, libraries, user manuals, and other information sources are customarily created either by subject matter experts or by professional indexers.

We have collected some data relevant to this issue in the recipe study. Our key-word providers represented several levels of expertise. The situation is not unrepresentative; usually the indexes of cookbooks and recipe collections are created by cooking experts, presumably on the assumption that their characterization and labeling will be superior, even for less sophisticated users. We calculated repeat rates separately for the three groups of key-word providers (experts, intermediates, and novices) subdivided by whether they had produced the key words under instructions to make them appropriate to novices or to experts. The results are shown in Table II. The repeat rates shown were calculated in a special way. For each cell, the index words were provided by particular subsets of describers, and the proportion of matches was calculated on a pool of descriptor words provided by other subjects. Thus, the (ee, ee) cell estimates how likely a word provided by one expert for other experts was to match that provided by another expert with the same, expert, audience in mind. The (en, ne) cell estimates the probability that a word provided by an expert

* In intermediate cases, like program and command names for an operating system, the method may be satisfactory for expert users, while leading to dissatisfaction for others (see Refs. 11 and 12).

Table II—Repeat rate measures of agreement between indexers and users for the recipe data set

Group	ee	en	ie	in	ne	nn
ee	.11	.17	.14	.15	.10	.22
en	.17	.11	.18	.17	.16	.21
ie	.14	.18	.20	.20	.21	.23
in	.15	.17	.20	.19	.17	.26
ne	.10	.16	.21	.17	.16	.16
nn	.22	.21	.23	.26	.16	.32

for novice users would match the word provided by a novice for expert users. Clearly, the differences among marginal values (i.e., the averages of repeat rates for different index providers) are not large, and more clearly still, expert cooks do not provide better descriptions for the use of either other experts or novices. (If anything, novices do the best in using each other's words.)

The usual armchair approach, even if undertaken by subject matter experts, has only a small rate of success. The obvious step at this point is to seek explicitly optimal choices of names, treating this as the empirical question that it clearly is.

4.1.4.3 Version: optimized (best). If we want to use the name that has the greatest currency among subjects, we must choose the term that is in fact maximally used by subjects for each object. We pick the maximum cell in each column and use the corresponding term.

Evaluation: by various estimates of the range of expected performance. The lower bound is based on split halves analysis; the upper bound is based on a transformation of the column repeat rate and from an analysis that assumes that the sample data exactly reflect the population probabilities.

To know the performance of this model we need to know the true population magnitude of the maximum cell in a column. That cell is the one we would choose according to an optimum name assignment scheme, and its size would be the proportion of future users' terms for the given object that would coincide with our optimal choice. Problems arise in that there are no known distribution-free, unbiased estimators of the population magnitude of the maximum probability cell. We have, however, been able to devise a few techniques that let us put bounds on the performance of this system. The first uses a split-halves technique to give a lower bound estimate. The data in each column are split into two halves, and the maximum cell chosen on the basis of the first half (as though we had to design our "optimal" system on the basis of half the data). This cell is then matched against the second half to see how well it succeeds. Thus, the second half acts as a virtual experimental test of the performance of the "optimal" method. The split-half results shown here are the average performance of ten independent splits of each data set. See Results Table 3.

Results Table 3

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.15	.19	.26	.26	.31

These numbers are an unbiased estimate of how well a system would do using this optimum strategy but constrained to a small amount of data (namely the amount in each half). It clearly underestimates how well one could do with more data. We have used two approaches to obtaining an upper bound. One is based on an interesting inequality relation that holds between the size of the maximum cell and the repeat rate statistic described above: It can be shown that the former is no greater than the square root of the latter.*

Thus, the performance of the optimum model is expected to be no greater than the square root of the performance of the weighted random (armchair) model. This statistic will overestimate performance to the degree that individual words other than the maximal one are also applied frequently to a given object. Since our own data suggest that this is commonly true, this upper bound is likely to be quite generous. It has an important pragmatic advantage, however, in that it is independent of sample size and easy to obtain. Other estimates of optimal performance require collecting detailed data on the precise pattern of naming. This estimate requires only that one observe the probability that two people use the same name (i.e., the repeat rate), without even having to note what particular terms they use. Taking the square root then yields an upper bound estimate for the best possible single name per object. For our data, these quantities are listed in Results Table 4.

Results Table 4

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.27	.32	.33	.35	.42

The final way to estimate the performance is to let the data predict itself. The observed largest proportion falling in a cell is taken as the estimate of the population maximum. A familiar result in rank order

* This relation follows from two inequalities. First note that the repeat rate is the sum of the squared cell probabilities. This sum is clearly greater than or equal to any of its terms, since all are positive. In particular, it is larger than the square of the maximum probability cell. Thus, the expectation of the repeat rate is larger than the expectation of the squared maximum cell. Second, note that the expectation of any squared variable is always greater than or equal to the square of that variable's expectation. Putting these together, the expectation of the repeat rate is greater than or equal to the square of the expected magnitude of the maximum cell. Thus, an upper bound on the maximum cell is estimated by the square root of the repeat rate.

statistics is that in the presence of error the observed maximum of a number of observations is expected to be larger than it should be. Thus, using the maximum to estimate itself is likely to be an overestimate for any limited sample size. This is particularly true for small samples. In the extreme case note that if a column of cells has only one observation, the observed maximum will be in the one cell where the single observation fell, which will have an estimated probability of 1.0, regardless of the true underlying probabilities. This will become more relevant later when we calculate similar statistics on rows of the matrix, where many of them will involve small numbers of total observations in each row. For now, however, the samples are not too small, and the results are presented in Results Table 5.

Results Table 5

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.16	.22	.28	.34	.36

4.1.5 Discussion

It appears that performance would be about twice as good when using an optimum naming strategy than when using the weighted random model (which we believe to be an approximation to typical current practice) in a one-word-per-object system. Still, the overall levels are not very impressive. It should be noted that these optimal strategies represent the best *any* single-name scheme could do. No expert, human or otherwise, could choose single names that would work better.

Is this the best that we can hope to do for people? The answer is no. There are other, more effective approaches. One is to use multiple names, sometimes called "aliases", for each object, which leads us to our second general structure for a system model.

4.2 Model 2: "Several names per object."

4.2.1 Interpretation

Each object in the system is assigned M terms or names. The user enters one term. Success depends on the user's word coinciding with any one of the system's M words.

4.2.2 Motivations

Giving things single names is not adequate for the uninitiated, so a reasonable next step is to try giving the system several names by which to recognize each object.

4.2.3 Structure

The constraint is that exactly M words are stored for each object.

4.2.4 Analyses

4.2.4.1 Version: random. For each object M names are chosen randomly from the total vocabulary, i.e., M cells are chosen from each column. The recall probability is M/R, and in our sample tables would range from .01 to .04 for three names per object ($M = 3$).

4.2.4.2 Version: weighted random. The first name is chosen with a probability proportional to its frequency in the given object's column of the table. Each successive name is then similarly chosen, without replacement, from the remaining cells. This is as though someone "sitting in an armchair" thought up several distinct names, any one of which the user could use with success. We make an independence assumption that the probability of a word being chosen is independent, except for renormalization, of the words already chosen. This is probably a faulty approximation for a single human generating a series of words, where the first word thought of may influence the next. If the words were separately proposed by different designers, this independence assumption might be more nearly correct.

Note that by reversing the roles of the system and the user we obtain a very interesting dual interpretation of this model. That is, let the system store a single word and the user make M different guesses. Success would be counted if any of the user's M words matches the system word. Either of these interpretations merely involves the probability that a single sample from the column will match one of M samples drawn without replacement from the cells of the same column (that is, without replacing the whole cell, not just the observation from the cell).

Evaluation: by "M-order repeat rate" statistic, within columns. These success probabilities can be estimated using an extension of the repeat rate statistic to this case of multiple samplings without replacement of types. The probability for column j can be shown to be estimated by:

$$\hat{r}_j^{[m]} = \sum_{i_1} \frac{n_{i_1 j}}{N} \frac{n_{i_1 j} - 1}{N - 1} \left[1 + \sum_{i_2 \neq i_1} \frac{n_{i_2 j}}{N - n_{i_2 j} - 1} \left[1 + \sum_{i_3 \neq i_1, i_2} \frac{n_{i_3 j}}{N - n_{i_2 j} - n_{i_3 j} - 1} \left[1 + \cdots \right. \right. \right.$$

$$\left. \left. \left. \cdots \sum_{i_m \neq i_1, i_2, \ldots, i_{m-1}} \frac{n_{i_m j}}{N - \left(\sum_{k=2}^{m-1} n_{i_k j} \right) - 1} \left[1 \right] \cdots \right] \right] \right].$$

This formula requires a calculation time that grows exponentially in M. We therefore limit results to $M \leq 3$. See Results Table 6.

Results Table 6

recall probability	Ed5	Ed25	CmOb	Swap	Recp
($M = 1$)	.07	.11	.12	.14	.18
($M = 2$)	.14	.21	.21	.25	.33
($M = 3$)	.21	.30	.28	.34	.45

Note that allowing the system (or, equivalently, the user) several words for each object in trying to match its partner achieves considerable gain. Performance almost doubles and triples as we go to two and and three guesses.

It should be remembered that the estimated probabilities of success for 1, 2, and 3 guesses were calculated on data from subjects' first responses only. Under the interpretation of this model in which subjects make repeated guesses at a system's single word, this is equivalent to assuming that successive guesses, given that previous guesses had failed, would resemble other first-provided words. To get a true estimate of the expected performance of a system that actually used this technique, we would need data on people actually making successive guesses. One would have to know how many such guesses can actually be made and with what quality before one could know what the maximum performance would be. However, if one supposes that the system, or perhaps the users' desires and abilities, limited input to three guesses, performance of such a system would not be likely to exceed about one chance in three of correct return.

4.2.4.3 Version: optimized. Choose as the M names for each object those terms that are maximally used by subjects for the object; that is, pick the highest M cells in each column and use the corresponding terms. This is a simple generalization of the single-name case.

Evaluation: by split halves and self-estimation. We again run into the problems involved in estimating maximum, and now also the nearly maximum, cells. The split-half and self-estimation procedures were used here to estimate upper and lower bounds. The split-half results represent the average performance of ten independent splits of each data set. The results are presented in Results Table 7.

Results Table 7

recall probability	Ed5	Ed25	CmOb	Swap	Recp	
($M = 1$)	.15	.19	.26	.26	.31	(split half)
	.16	.22	.28	.34	.36	(self-estimation)
($M = 2$)	.26	.32	.36	.36	.49	(split half)
	.28	.37	.41	.50	.56	(self-estimation)
($M = 3$)	.37	.42	.42	.45	.58	(split half)
	.38	.49	.48	.59	.67	(self-estimation)

4.2.5 Discussion

Considerable benefit is obtained both from using data to optimize choices and from giving the system several names for each object (or, equivalently, allowing the user several guesses). Though it was not undertaken here, it would be interesting to explore the possibility of letting both the user and the system use several names to try to match each other.

The improvement gained by storing multiple words has associated with it a potential cost in ambiguity. Nowhere in either this model or the previous, one-name model has there been any concern that the names be distinct. The names for each object were picked from the corresponding column, with no regard for what names were being chosen for other objects. This means that two objects could be assigned the same name, with consequent ambiguity should the user give that name. The system would be unable to tell which object the user intended, and would have to present the user with a menu of choices to differentiate among them. (Recall that at the outset we stated that we would be assuming such a system, and, moreover, that choices among the items on the subsequent "menu-on-the-fly" would be assumed error free.)

Naming choices will collide when two different objects have the same words applied to them. One might expect this to happen, for example, if two objects are very similar, so that the same words apply to them. We have collected some data that confirm the existence of this similarity effect. For both the recipe and the common object data we compared the probability of a naming collision for random and similar objects in the set. Subjects were given 5 "focal" objects and 25 "match" objects, all drawn at random from the set. For each of the focal objects, they were told to select the one match object that was most similar to it. Using the weighted random naming method, the naming collision rates were compared for these five pairs of similar objects and the same five focal objects paired with random members of the match set. That is, we calculated the probability that a word applied by one user to the first object would be the same as the word applied by another user to a second object. Using subject differences as a random error estimate, there was a highly significant increase in naming collisions for the pairs of objects judged to be similar [$t(14) = 3.86$ and $t(24) = 5.37$ for the recipe and common object data, respectively].

To examine the effects of trying to avoid collisions, we studied the next model.

4.3 Model 3: "A distinct name for each object."

4.3.1 Interpretation

Each object in the system is assigned a single term, with the proviso

that no term may be used more than once. Success depends on the user's spontaneously produced word coinciding with the system's distinct name for the intended object.

4.3.2 Motivations

Note that in typical naming circumstances the intent is often to establish conventions about terminology so as to avoid the ambiguity that would otherwise arise. Naming conventions are used not only to set by fiat the name by which an object will be known to a system, but also to proclaim that *only* that object will be so known. Often interaction will reach a stage where complete precision is needed, e.g., for actual command execution.

It may be the designer's motivation in selecting names that users learn terminology that allows them to be precise. As had already been pointed out, however, the designer's intent may not correspond with the user's reality; the untutored may try to deal with the system anyway. Thus, we explore the user-guess success for systems designed with the unique name constraint.

The models discussed here are just like those of the "one-name" case, except that words cannot be used more than once.

4.3.3 Structure

One distinct word is stored for each object, i.e., this is the same as Model 1, except that the words must all be distinct, so that at most one object is referenced by each word. This imposes strong constraints on the system table, on both row and column totals, and the number of rows and columns used overall. This high degree of constraint makes mathematical treatment difficult, as will be discussed shortly.

4.3.4 Analyses

4.3.4.1 Version: random. For each object, one name is chosen randomly from the total vocabulary. Once a vocabulary item is used, however, it is eliminated from any future consideration.

Recall probability would be $1/R$ and range from .002 to .014 for these data. The "number returned" is easy to give for this case, as it is set by the structural constraint, that each name have a unique referent. It is therefore 1 for all models having this structure. That is, when the system finds a target it returns exactly one candidate. However, there are many occasions when the user word matches no system word, and so no target is returned.

4.3.4.2 Version: weighted random. Here the name is chosen with a probability proportional to its frequency in the given object's column of the table. Then the chosen word is eliminated, and a name is chosen in an analogous way for another object from the remaining words, etc.

The results are clearly influenced by the sequence in which objects are dealt with. The approach used here was to choose in the following way: a word/object pair is chosen by a weighted random sampling from the whole table. This gives the right distribution within each column and allows appropriate representation of each column. Once such a cell is chosen, the corresponding object has been named and the word used, so both are eliminated, and the procedure is iterated on the table, now reduced by one row and column, until all objects have been named.

Evaluation: by Monte Carlo simulations with split halves. We could devise no way to evaluate this model analytically, so we used a Monte Carlo simulation on split halves. We divided the data in the user table in half and randomly picked names, according to the model outlined above, using the data from one half. The second half of the data was then used to evaluate the effectiveness of the names thus chosen. The results presented here are the average for ten split halves with ten independent Monte Carlo simulations of name selection in each. See Results Table 8.

Results Table 8

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.07	.08	.11	.12	.09
number returned = 1.0 (by design)					

Note that, as might be expected, success is less than for the comparable model in which names did not have to be distinct. In some instances this decrease is small, as with the edit studies; in others it is large, as with the recipes. This should depend on whether there were a few high-frequency words that were used for many different objects.

4.3.4.3 Analysis of precision versus popularity of terms. In all of our data sets there was a slight trend for high-frequency words to be less discriminating than low-frequency words. In studying this relation quantitatively, the measure of discriminating power of a word was given by the repeat rate statistic, this time applied within each row. This row-repeat rate is a measure of the probability that two uses of a word refer to the same object. When this probability is high, the word is very discriminating. Below we present the correlation (Spearman r) of row-repeat rates with the marginal frequency of the words. See Correlations Table 9.

Correlations Table 9

	Ed5	Ed25	CmOb	Swap	Recp
corr:	−.28	−.30	−.21	−.24	−.16
(N words):	(74)	(84)	(301)	(145)	(167)

This correlation points to the cause of the difficulties run into by the constraint of distinct names. There is something of a conflict. If one chooses high-frequency names, they will be likely to collide. If one chooses less frequent names, the chances of collision will be somewhat less, but fewer users' queries will be handled. Unfortunately, there is no correspondence between the size of these correlations and the size of the decrease in performance for each data set. The reasons for this are not clear.

4.3.4.4 Version: optimized (though not best possible). The idea is to choose the distinct names such that the total probability in the cells chosen is maximal. The method used here is a greedy algorithm that is not truly optimal, but it is a reasonable improvement over the weighted random method. It begins by picking the highest cell in the matrix; then, after eliminating the corresponding row and column from the matrix, it iterates, picking the next highest cell, etc., until all objects have been assigned a name. Algorithms like this are called "greedy" because at each step they take the biggest possible chunk of what is left, without regard for what later problems that may cause. In this case, it is possible that an early choice will eliminate a word that would be very good for another object, when there was an alternate word that would have done almost as well at no such cost. Algorithms that would take such future complications into consideration, or equivalently consider so many possibilities at once, are typically combinatorially explosive. Thus we present the results of the straightforward greedy algorithm approach.

Evaluation: by split halves. There is no simple method to estimate the performance of such an approach. Again we turned to the split-halves technique of dividing the data in half, applying the algorithm to one half and then testing the chosen names on the other half. The results, averaged over ten independent splits, are given in Results Table 10.

Results Table 10

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.14	.11	.23	.19	.11
number returned = 1.0 (by design)					

4.3.5 Discussion

As we expected again, the optimization attempt, even though imperfect, has a dramatic effect, in many cases doubling the performance of the system. Even in the best case, however, the system succeeds only about one quarter of the time, and typically little more than a tenth of the time. We note that the numbers here are substantially lower than in the case where there was no requirement that names be distinct. In fact this improved method really does no better than an

armchair (weighted random) version of the unconstrained model, and often worse. The lesson here is that adding the requirement of uniqueness, common in establishing conventions, hurts naive users quite a bit. The need for such conventions is not denied, it is simply asserted that auxiliary aids will be needed for systems that make heavy use of such conventions. It is worth noting that, for our data sets, the number of objects that are not disambiguated and would have to be presented for a second stage choice is always small, and the gain in recall always large. Thus, the "menu-on-the-fly" method implicit in several of the models presented here appears to be very promising as an aid to unsophisticated users.

4.4 Model 4: "Distinct names, augmented with M extra referents."

4.4.1 Interpretation

Each object is given a distinct name. $M - 1$ other referents for those words are also stored.

4.4.2 Motivations

Part of the inferiority of the distinct-name models, when compared to the unconstrained models, came from the fact that people often want to use the same names to refer to several objects. The motivation for the next model is to recapture access to some of those other interpretations of the term. A simple extension of the distinct-name structure is to begin with the situation where each object is associated with a distinct name, one that can be memorized and used unambiguously by experts, but also to store $M - 1$ other objects that the term applies to, explicitly for use in a "menu-on-the-fly" for the untutored. A reasonable system implementation might be to give the "distinct" name a special status (the "real" meaning of the term), and to admit the other interpretations as secondary, perhaps to be verified in the context of use.

This is the first model in which we explicitly design in more than a single system guess as to the intended object. To be sure, several guesses for the meaning of a term could (and would) have arisen in the uncontrolled name cases of Models 1 and 2, but here we predetermine exactly how many guesses the system makes and evaluate performance as a function of this parameter.

As more objects are returned, the likelihood of including the intended target is increased, but a certain cost is incurred—the cost associated with discerning the true target among all the returned objects. Data searches can fail not only by not giving access to a desired object, but also by returning too many unwanted objects. In information science the problem is familiar as the trade-off between *recall*—the number of desirable items returned—and *precision*—the

proportion of items returned that are desirable. This is also similar to the hit versus false alarm trade-offs of signal detection, familiar to psychologists and communication theorists. In the latter context the trade-off is sometimes examined by tracing out operating characteristics and interpreting them in terms of parameters of an underlying statistical theory. There does not currently exist a corresponding statistical theory for our precision and recall rates, but it is useful to be able to examine, or in this case, control, precision explicitly and see what gains in recall result. This yields an operating characteristic. Any version of the general baseline model in which the system is set up to return an explicit number of guesses provides a direct way to do this.

4.4.3 Structure

At least one name is given to each object, but each name that is used by the system also refers to M different system objects.

4.4.4 Analyses

4.4.4.1 Version: random. Here the primary cells are again chosen completely blindly, eliminating rows and columns already used in an iterative manner exactly as in the distinct-name case. After the C primary cells are chosen, the $M - 1$ additional or secondary cells are chosen randomly from each new row chosen.

The resulting probablity is M/R and would range from .001 to .035. The number returned is, of course, M by design for all versions of Model 4.

4.4.4.2 Version: weighted random. Cells are chosen at each step with a probability reflecting their magnitudes. Thus, first C distinct names are picked, as in the distinct-name model (Model 3). Then, in the row of each cell just picked, $M - 1$ more objects are chosen with a probability equal to their relative frequencies in the rows.

Again, the weighted random case is an approximation to what an individual designer might do without collecting data from other people. The process of coming up with additional interpretations requires a bit of further explanation. The asymmetries found in free association data suggest that people starting from terms and generating objects would yield probabilities quite different from those obtained from people starting with objects and generating terms.[13] This asymmetry means, in this model, that the additional interpretations cannot be thought of as the result of the designer sitting in an armchair and thinking up other interpretations for the terms. The efficacy of such an approach is not estimable from our data.

A scenario that might better satisfy the assumptions of this model would have the system memorize the first $M - 1$ nonstandard uses of its known terms that it comes across.

Evaluation: by Monte Carlo simulation on split halves. The data are given in Results Table 11.

Results Table 11

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =					
($M = 1$)	.07	.08	.11	.13	.08
($M = 2$)	.13	.15	.15	.17	.15
($M = 3$)	.18	.21	.18	.20	.20

The number of guesses that the system is requiring to achieve this performance is of course, M, the number returned by design.

4.4.4.3 Version: optimized (though not best possible). Again it was not feasible to find the completely optimal choice of cells. Instead, the primary cells were found in the same "greedy" way used for the simple distinct-name "optimal" case. The subsequent choice of secondary referents for the words so selected can be optimized by choosing the $M - 1$ largest cells remaining in the row. Note that it is possible for some of these remaining cells to be larger than the primary cell, as when they refer to an object that was eliminated before the row was chosen. (It is the vagaries of the need for convention in names that brings about this ironic use of the term "primary.")

Evaluated: by split halves. In the now familiar procedure, the data are split and the first half used to choose the cells following the algorithm just outlined. Then the second half of the data is used to evaluate the choice of cells. See Results Table 12.

Results Table 12

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =					
($M = 1$)	.14	.11	.22	.21	.11
($M = 2$)	.21	.20	.28	.27	.17
($M = 3$)	.27	.29	.31	.30	.22
number returned = M (by design)					

These results, for one to three total guesses, are plotted in Fig. 2. The model curves can be viewed as operating characteristics, giving total recall as a function of decreasing precision. The dashed diagonal lines represent expected chance performance if the system returns guesses at random. Note that in the case of the Ed5 data, the random performance is *better* than that of this model. This is possible because this model, like most current computer systems, makes no response at all to any but a few words. By failing on so many of the words it encounters, it can be surpassed by a system that only guesses, but at least guesses for all words. Admittedly, in the context of command execution any level of guessing without confirmation may be danger-

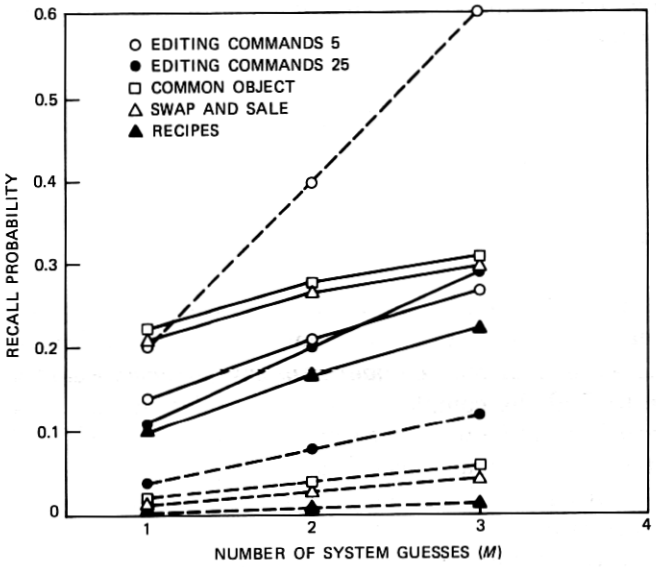

Fig. 2—Recall as a function of the number of system guesses for Model 4 ('distinct names, augmented with M extra referents') is shown by the solid lines. The dashed lines represent expected chance performance if the system returns guesses at random. (Different chance performance for the different data sets simply reflects the different number of objects in each set.)

ously out of place, but there are safer arenas, like help facilities, that could benefit. In any case the cost of ignoring so many user words is made clear. Notice that the total probability does not approach one, even in the case of command verbs for editing operations where there were only five objects. The reason is again that many words were used by our subjects to describe these objects, and the algorithm picked only a maximum of C of them to recognize. The modeled assumption was that if a subject supplied any word not recognized by the system, no choices would be returned and a failure would result.

4.4.5 Discussion

These models use the same number of cells as the multiple-names models, yet do not do as well. The implication, presumably, is that there is considerable cost in limiting oneself to a small number of words. The users distribute their descriptors too broadly for any such limitation to be very satisfactory.

Thus we are motivated to try a completely different approach. All the models up to this point have inherently focused on what the system brings to the interaction. The modus operandi has been to set all the system's objects before us and then try various ways to guess what users will call them, albeit with some improvements as we give

deference to the empirical characteristics of the user's language as seen in the user table.

Suppose we turn the tables, so to speak, and focus on what the users bring to the interaction: the terms they use when trying to specify things. Start with the terms, all of them, and concentrate on trying to guess to what object they refer. This brings us to our next model.

4.5 Model 5: "Recognize one referent for every word."

4.5.1 Interpretation

The system stores every word that it can, and for each has one guess as to the intended object.

4.5.2 Motivations

One of the principal problems with the approaches studied so far has been that a large proportion of the recall failures can be attributed simply to not recognizing many of the user's words. The systems modeled have paid too little attention to the wide variety of the terms people use spontaneously. Here we focus on this essential character of what the users bring to the interaction. For every word produced during the collection of the data for the user table, a guess is made. The different versions vary as to how these guesses are made.

It is useful to note that these models, and their generalizations in the next section, are really duals of the single- and multiple-names models. In those models the system had words for each object, and it succeeded when its name was the same as the word the user would use. Thus, to evaluate those models we calculated the conditional probability that, given a target object, the system and user names would coincide. Then we summed probabilities across objects, weighting each by its column's marginal probability (all essentially equal in our data, by design). Here we conditionalize the other way. Given the word used, we find the probabilities that the system and user associate the same object with the word. Then we sum across words (rows), weighting by the very uneven observed marginal frequencies for each word, to get the overall performance.

4.5.3 Structure

The system makes use of every word in the table, associating each with one, and only one, system object.

4.5.4 Analyses

4.5.4.1 Version: random. Here the system will associate random objects with the input words. That is, the system will simply make a

random guess for anything the user says. The recall probability is just $1/C$, and ranges from .005 for the recipes to .200 for the Ed5 data. The number of guesses being made to achieve these performances is predetermined by design to be 1 for all these models.

4.5.4.2 Version: weighted random. Here an object is associated with each word by a random method that takes each object with probability proportional to its relative frequency in the row. Arguments similar to those made for the augmented distinct-name model here imply that this is not the analog of having the designer sit down with the list of words and hazard guesses as to what they mean; the data used here go in the other direction. The more appropriate scenario is as if a system records the first encounter with a new word, and its intended referent, and makes a single pointer based on this single observation, forever freezing the meaning of the word. This is perhaps an unreasonable scenario, but the model is worth investigating because the statistics that result have other valuable interpretations.

Evaluation: by row-repeat rate statistic. The probability of success in this case is the probability that two users will mean the same thing by a given word. The results below are the average row-repeat rates, weighted by total row frequency. (Rows with a frequency of one were excluded, since the repeat rate is not calculable in that case.) This number is an unbiased estimate of the overall probability that any two occurrences of any word will be in reference to the same object. See Results Table 13.

Results Table 13

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =	.41	.15	.52	.62	.13
number returned =1.0 (by design)					

We note how variable the probabilities are. They are low where the same words mean different things, and high where any one word refers to only one object. Apparently these domains differ in this aspect. There are at least two possible explanations. First is the similarity effect discussed in connection with Model 3. There it was shown that pairs of objects judged to be similar had higher overlap in the patterns of names applied to them than did random pairs. The resulting naming collisions meant that it was more difficult to find distinct names for similar pairs. Here we are explicitly interested in a different, though related, effect. We can use the previous experimental data (see Section 4.3.4.3) again, this time to demonstrate the similarity effect on a row-repeat rate measure. In this version we consider pairs of cells in each row: one cell from the target column, and one from either a column

rated similar or from a random control column. The repeat rate is calculated on the two cells and summed (weighted by row sum) down all the rows. Since a low repeat rate for a given word indicates that it does not discriminate well between the objects, we predict a lower repeat rate for the similar pairs of objects, and that is what we obtain. In the recipe data the mean repeat rate of the similar pairs was .73 and for the random pairs was .91 [$t(14) = -5.83$]. For the common object data, the similar pair repeat rate was .89 and the random repeat rate was .95 [$t(24) = -5.71$]. Thus similarity has a strong effect on repeat rate. Unfortunately, what we need to make sense of the varied results is some measure of the relative internal similarities of the various domains. Such data are neither available nor easily obtained. Still, it might be agreed that the set of common objects is more diverse than is a set of cooking recipes, or text editor operations, corresponding to observed differences in average repeat rates.

There is another factor that might be helping the swap-and-sale descriptors. Analysis of the other data sets was either strictly limited to single words, or else only to short phrases with few content words. Swap-and-sale descriptors contained an average of 1.7 content words, where there were fewer than 1.2 content words in the others. If these words tend to be used in a conjunctive sense and if they are not redundant, they should contribute to the high selectivity of the descriptors seen for the swap-and-sale data.

A final note should be made of the higher repeat rate for Ed5 compared to Ed25. Part of this is due simply to the fact that even with a purely random, undiscriminating distribution of name usage across objects, there would be an increase in repeat rate as the number of objects is decreased. If there are only two objects, people will have to mean the same object by any given term at least half the time. A more intriguing possibility is that the classes are more distinct entities than are the individual objects, and that this makes word usage more discriminating, above and beyond the chance effect just mentioned.

4.5.4.3 Version: optimized. In this case the user matrix is used to find the best possible choice for a word's referent. This is done by picking the maximum cell from each row, the object to which the word has most commonly referred.

Evaluation: by split halves, square root of the row-repeat rate, and self-prediction. Evaluation is again problematic. While there is no question that picking the maximum cell is the best possible strategy, there is no unbiased estimate of its true magnitude. Thus we resort to the same three methods used when similarly confronted in the optimal version of the one-name model (Model 1): the split-half technique, an unbiased estimate of how well one could expect to do with half the data; and two estimates of upper bounds. See Results Table 14.

Results Table 14

	Ed5	Ed25	CmOb	Swap	Recp	
recall probability =						
	.49	.18	.43	.35	.13	(split half)
	.62	.35	.65	.72	.28	(square root of row-repeat rate)
	.54	.26	.69	.81	.25	(self-prediction)
number returned =1.0 (by design)						

4.5.5 Discussion

There is considerable variability, both in the performance and in the range of these estimates. The largest range is for the swap-and-sale estimates. This is due to the very large number of descriptors that occurred only a few times. The data for each such descriptor are statistically unreliable, so the maximum cell cannot be accurately identified; this problem is severely aggravated by splitting the data and only using half to make the identifications. Note that this is not an experimental artifact. A very long tail on a descriptor distribution is a legitimate real-world problem, because it means that new terms will keep arising that the system will not have seen before, and about which it will not be able to make any educated guesses. A wide range in the estimate reflects the fact that very large amounts of data would be needed to approach asymptotic performance.

As for the overall diversity of scores from domain to domain, the arguments about similarity and number of words in the descriptors, given above for the repeat-rate case, are equally applicable here.

The most important point to be made about this optimized model is that it represents the best one could possibly do. We can do no better than to recognize every word users use and make an optimal guess as to its meaning. Clearly, if we have insufficient data to do this well, either preventing us from recognizing a word, or from being able to estimate the modal referent, performance will suffer. But the upper bound estimates represent the real, strict limits of performance. No other pattern of structural constraints could do better, except at the cost of precision. This trade-off with precision is explored in our final model, in which an explicitly prespecified number of multiple guesses is returned to increase the chance of including the user's intended object among them.

4.6 Model 6: "M referents for every word."

4.6.1 Interpretation

The system stores every word that it can, together with a set of M possible referents. Whenever the user enters a word, the system returns the M guesses, possibly ranked in some way.

4.6.2 Motivations

The "one referent for every word" model (Model 5), in its optimized version, gave the best possible performance for a system that hazards only one guess for a user's word. The only way to increase the chance of returning the user's intended object is to return more than a single guess. These guesses could be returned in the form of a menu of M items, among which users would choose.

As we mentioned, this is the dual of the multiple-name set of models, and several of the evaluation procedures differ only in that rows have traded roles with columns.

4.6.3 Structure

Like the previous model, the system makes use of every word in the table, but here it associates M system objects with each.

4.6.4 Analyses

4.6.4.1 Version: random. The system makes just M pure guesses, without replacement, from the set of system objects. The recall probability for these cases is thus exactly M/C, and the number returned is M, by design, for all versions of Model 6.

4.6.4.2 Version: weighted random. In a manner following the single referent weighted random model, the M cells are chosen with a probability that is proportional to their relative sizes. The M choices are required to be distinct, and so a cell is excluded from further consideration once it has been selected.

The scenario that this might correspond to would be one in which the system learns the first M distinct referents of the word that it comes across in use. Its encounters would be governed by exactly these probabilities.

Evaluation: by M-order repeat rate statistic, within rows. We want the probability that the user's single intended referent coincides with any of the system's candidates, when both sets are drawn from the same probability matrix. These success probabilities are estimated using the same extension of the repeat rate statistic used in evaluating success of the many-names-for-one-object model (Model 2, Section 4.2.4.2). Here though, it is applied to the cells within a row, rather than within a column.

It should be noted that the formula requires there to be M nonempty cells in the row, to prevent some of the denominators from going to zero. For example, it is not possible to give an unbiased estimate of how well three guesses would do if the data only show two objects. Note that while rows with highly discriminating words should in general have high M-order repeat rates, it is exactly such rows that will be less likely to satisfy this requirement, especially at lower

marginal frequencies. Thus, ignoring rows where this statistic cannot be calculated biases the average value of this statistic downward. Though it is not clear how to get an upper estimate, there is another way to get a lower bound, by recognizing that performance for M guesses is at least as good as performance for M-1 guesses. Thus, a lower bound on the average performance comes from using for each row the calculable repeat rate of highest order not greater than M. See Results Table 15.

Results Table 15

	Ed5	Ed25	CmOb	Swap	Recp
recall probability =					
($M = 1$)	.41	.15	.52	.62	.13
($M = 2$)	.66	.26	.65	.74	.21
($M = 3$)	.81	.36	.71	.80	.27

4.6.4.3 Version: optimized. This version takes the optimal strategy for multiple guesses. The user gives one word, and the system makes guesses that the user means one of the M most likely things the word has meant in the past, as deduced from the data in the user table. That is, it returns the objects associated with the M highest cells in the row corresponding to the word given. (A sequential version of the model would give the user the guesses in decreasing observed frequency, beginning with the maximum cell.)

Evaluation: by self-estimation and split halves. As in other cases where evaluation of performance depends on the estimation of true population ordered frequencies, we must resort to indirect methods to give a range. We use split halves to give a lower bound and self-estimate to give an upper bound. See Results Table 16.

Results Table 16

	Ed5	Ed25	CmOb	Swap	Recp	
recall probability =						
($M = 1$)	.49	.18	.43	.35	.13	(split half)
	.54	.26	.69	.81	.25	(self-estimation)
($M = 2$)	.70	.30	.53	.43	.20	(split half)
	.76	.42	.82	.92	.37	(self-estimation)
($M = 3$)	.83	.40	.58	.47	.26	(split half)
	.88	.53	.88	.96	.46	(self-estimation)
number returned = M (by design)						

The recall rates, as estimated by the conservative split-half procedure, are given in Fig. 3. The curves can be interpreted as operating characteristics; the distance between the curves and the corresponding dashed diagonal line reflects how much the implied system could be expected to outperform a simple menu system based on the pure random model given above, i.e., on a random choice of M objects. All

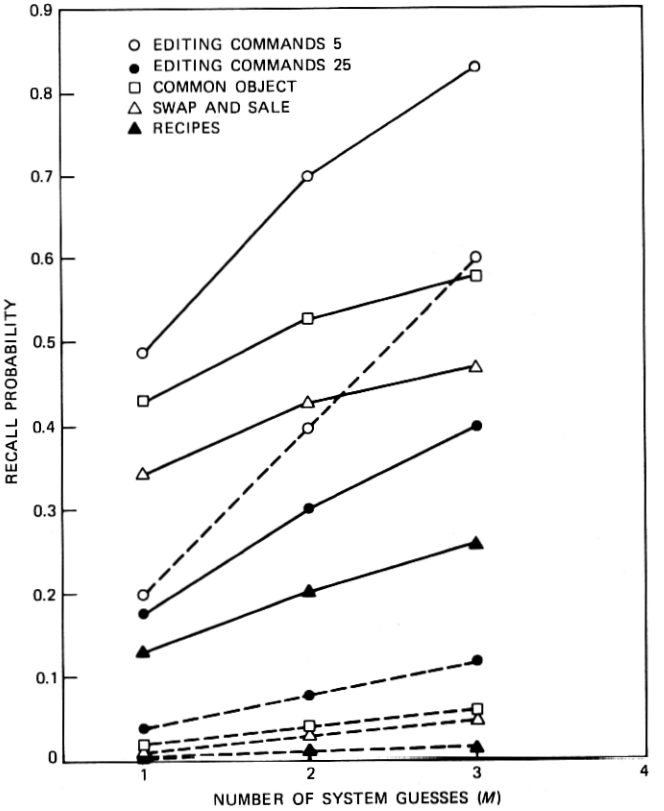

Fig. 3—Recall as a function of the number of system guesses for Model 6 ('*M* referents for every word') are shown by the solid lines (split-half estimation). The dashed lines represent expected chance performance if the system returns guesses at random. (Different chance performance for the different data sets simply reflects the different number of objects in each set.)

of our models do substantially better than chance. More importantly, they do much better than the augmented distinct-name model (Model 4) presented in Fig. 2.

4.6.5 Discussion

There are really two important points to be made about these results. The first is that these performances are variable and but moderately high. Note that the very high value of the three-system-guess case in Ed5 is rather vacuous, since there were only five objects to guess from. In the common object case, however, the high values are quite meaningful, since there were 50 objects.

The legitimate high values lead us to the second observation, that substantial improvement has been gained over the traditional, one-

distinct-name approach. The presumption is that this gain is due in part to the increase in capturing all the user's inputs. There are, after all, two ways for the system to fail in retrieving the user's target. One is by making wrong guesses, the other is by being unable to make any guesses at all. To give an idea of the size of this problem: for three of our data sets, no system using only C words (e.g., one for each object) could recognize even half the user's words. Thus, recall rates can only increase as the system is taught more words, and often this can be a sizable improvement. The only conceivable decrements (except those associated with time and space in index management) would be if the additional words were less precise, thus decreasing precision for a fixed recall rate, or decreasing recall for a fixed precision. But our finding of a consistent, albeit small, negative correlation between a word's frequency and its selectivity suggests that the opposite is true. As a greater number of lower-frequency words are included, precision will go up.

Recall also increases dramatically if the choice of algorithms is optimized, though it never exceeds the square root of the performance obtained by "armchairing", i.e., making pointers based on a single observation. The cost is in the trouble it may take to collect data. In some cases the variety of terms is so great that data collection must continue for quite a time before asymptotic performance is approached. Performance also increases as the system is allowed to make more guesses; this involves a direct trade of recall probability for reduced precision. In many circumstances, particularly in help facilities, this presents little problem. The user need only be given the various options and whatever additional information it takes to decide what is really sought.

4.7 Other possible models and some limits

The models presented so far have covered a large share of the space of simply constrained models where the user makes one guess, but they do not exhaust the ways in which a system could use empirical data on user descriptions to guess the user's desired objects.

One might consider, for instance, models that take advantage of redundancy in the tables. A good theory or description of what such tables are like could be used to augment the data to make better estimates of true population usage probabilities. For one example, if we knew that some single shape of distribution characterized all columns, we could use the data to estimate parameters of that distribution instead of individual cell probabilities. This would reduce the number of parameters that need to be estimated from the data, and consequently improve stability and reliability of predictions.

A related, and perhaps more interesting, idea would be to look for

latent structure in the similarity between objects, and between words, and use this to improve predictions. For example, if we knew or could infer from the data that two words were used almost identically, we might pool the cell frequencies for the two words; or if we knew that the objects and words related to each other by some more elaborate structure (e.g., a hierarchical tree), we could base predictions on calculated indirect reference paths.

While these ideas might be useful, the research presented in this paper puts strict limits on the success of any such approaches. As noted above in Models 5 and 6, the self-prediction evaluation procedure (determining the best guesses from the data and then using the same data to estimate success) and the "square root of repeat rate" evaluation procedure yield upper bounds on expected performance. Indeed, no analysis method of the kind we have just been discussing, no matter how clever, could improve on this result. The reason is as follows. Even using inherent structure to improve predictions could do no more than increase the accuracy of estimation of the population values for the input/output tables. Even with true population probabilities, however, the guessing decision rule would be the same; choose first the most probable referent of each word, and so forth. The input/output tables are inherently probabilistic, as a result of disagreement in word usage, not just of the estimation uncertainties. For a given user population, with a given degree of training, a certain amount of disagreement in word usage will occur. Even with perfect knowledge of the probabilities describing this usage, we could not predict individual referents perfectly. Now, the objective of having true population probabilities as cell entries is mimicked by our upper bound estimating procedure. It pretends that the values observed for each cell are exactly the probabilities that would exist in further sampling from the population. Thus, no method of "cooking" the data to reduce sampling error could achieve a better result than is illustrated by our upper bound values.

To improve performance beyond these limits, it would be necessary to construct systems that use different input, e.g., multiple words or interactive dialogues, or make the user learn to use more easily interpretable language. The last of these is the current default approach—make the user learn everything—and it has its limits for large systems or occasional users. Thus, we believe a more fruitful direction to explore further is multiple-word inputs.

The simplest models of multiple-word query might assume only content words (no syntax) and a statistical independence between words. Independence would allow the performance of such a system to be estimated from single-word data of the sort we have collected. Multiple words could be used disjunctively. In this case the system

would return any objects that matched any of the input words. This would result in an increase in probability of recall. Alternately, the words could be used conjunctively, so that the system would return only objects matching all the key words. The result would be higher precision. Mixtures of these two approaches (e.g., conjuncts of disjuncts) then clearly could be used to improve both precision and recall.

Of course, from our data we do not know how people can or do use multiple words, even without considering syntax. The popular balances of conjunction and disjunction are unknown, and spontaneous multiple guesses are no doubt not independent. But what sort of nonindependence is common? For example, do multiple terms tend to be more unrelated, or more redundant, than independence predicts? These questions must be explored before multiple-word systems can be theoretically evaluated, or optimized for the naive user.

The consideration of syntax leads to a whole new set of problems, those often encountered in Artificial Intelligence work on natural language understanding. We will not address them here.

4.8 Highlights

In the previous sections we used the tables of observed word usage to explore various models of how systems could name, and thereby give access to, their contents. We devised six general schemes for assigning names to objects. To test the schemes, we approximated user behavior by sampling appropriately from the tables.

Thus, for example, we simulated what word a designer might assign to an object (pick from the table), what word a user might choose (pick again from the table), and looked at how frequently the two words were the same. The results were taken to indicate how well a set of spontaneously generated single names for objects can be expected to work for untrained users. This particular example was called the "armchair model." The table is, after all, only a compendium of many people's spontaneous armchair attempts to give the best possible names for these objects. Thus, sampling from the table mimics designers picking names from their armchairs.

We used this general technique to explore variants on our basic model of interaction, in which we assume the designer builds in certain connections between names and the system's objects or services. These names were in some cases purely random, and in other cases were a simulation of a designer's best armchair guess; in still other cases the names were chosen more systematically, with a goal of optimality. In all cases, we assume that the user must "guess" the right name because we are concerned with untutored users who do not know the system words. The user's choice of words was always mimicked by sampling from the tables.

Of the models and variations, summarized in Appendices A and C, perhaps the most important are:

(1b) "One name per object: weighted random"
(1c) "One name per object: optimized"
(6c) "M referents for every word: optimized."

The first of these (1b) was the so-called "armchair" naming method just described. Expected results from the armchair method are shown in Fig. 4 model (1b). Note that despite the fact that the four domains differ dramatically in content, data collection style, and subject population, the results are quite consistent. They are all low. Two people—the designer and the user—will come up with the same name only about 15 percent of the time.

These results indicate that the current practice is deficient; it guarantees that it will be difficult for people to tell machines what

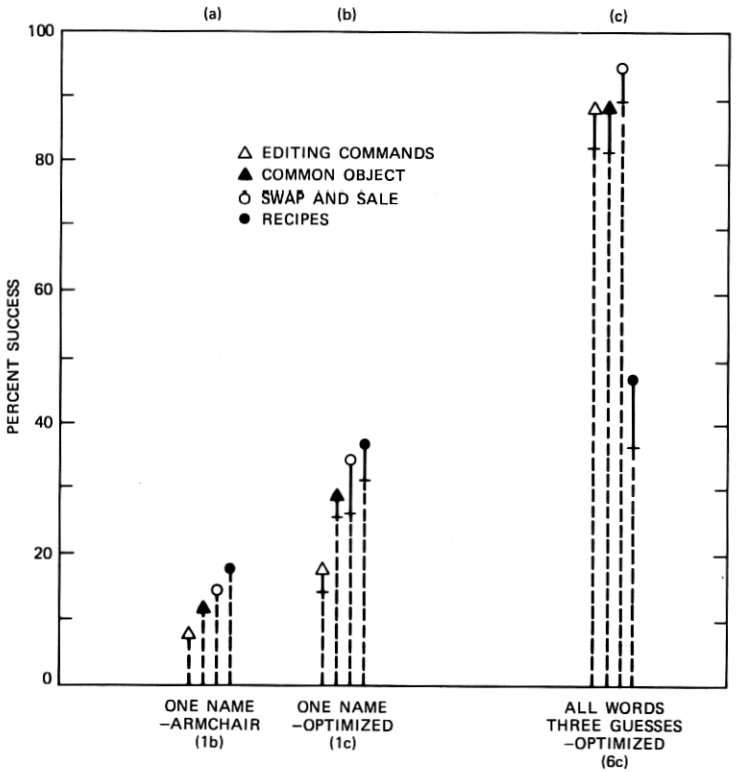

Fig. 4—Summary of expected success for three of the most important models presented. (a) "Armchair" model (1b); (b) Best possible single-name model (1c); (c) Model where the system recognizes all words and returns three guesses (6c).

they want, unless they already know what to say. Note that designers often have other, legitimate motives in assigning names—e.g., lack of ambiguity, memorability, or cuteness. But such naming practices only make the current problem worse; they lead to even less likely names.

The source of the problem is this. There are many names possible for any object, many ways to say the same thing about it, and many different things to say. Any one person thinks of only one or a few of the possibilities. Thus, designers are likely to think of names that few other people think of, not because they are perverse or stupid, but because everyone thinks of names that few others think of. Moreover, since any one person tends to think of only one or a few alternatives, it is not surprising that people greatly overrate the obviousness and adequacy of their own choices.

To improve performance of the "armchair" method we investigated whether experts in one of the content areas could pick better words. We had expert chefs choose key words for cooking recipes and found essentially no improvement: experts do not seem to do noticeably better picking terms from their armchairs than does anyone else (Section 4.1.4.2).

The next model of major interest (1c) substituted objective data for armchair suggestions. It used our data tables to identify the name that was most commonly used for each object. Thus, for example, for the operation we referred to as DELETE, the most commonly used term was "omit", with a frequency of 110, and that was the name used in this model. This empirical approach, occasionally advocated by human factors people, achieves the levels of expected success illustrated in Fig. 4 model (1c).

As discussed in Section 4.1.4.3, for statistical reasons we can only estimate certain upper and lower bounds on performance here. The lower one indicates how well one could do with a limited amount of empirical data from which to try to pick the best names. The upper one liberally estimates how well one could do with an infinite amount of data. The improvement over the armchair method is substantial, typically almost doubling the hit rate. This makes the point that the best name is a good bit better than the typical name. But it is also clear that even the best one is not very good.

It is critical tc note that these numbers represent the best one could possibly do by assigning a single name to each object. The problem is not solved by finding the right name. Different people, contexts, and motives give rise to so varied a list of names that no single name, no matter how well chosen, can do very well. Figure 5 is a plot of how many names are needed to account for a given percentage of the users' attempts, here for the common object data. (Here again we have the statistical estimation problem, so the boundary is double.) Note that

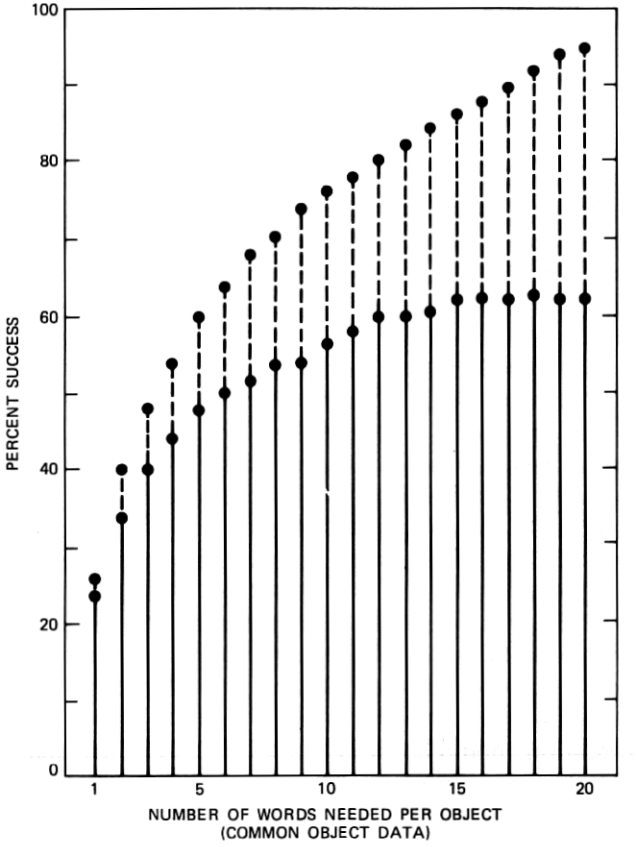

Fig. 5—A plot of how many different names (or "aliases") are needed for each object to account for a given percentage of success. This plot is based on data from the common object domain. The dashed lines indicate ranges.

even 15 words (word types or "aliases") per object account for only 60 to 80 percent of the words people apply. That is, even with 15 names stored per object, the computer will miss 20 to 40 percent of the time. The lesson clearly is that systems must recognize many, many names.

After considering several other models, we turned the problem around a bit. Instead of starting with system objects and looking for names, the new strategy was to begin with user's words and look for interpretations—that is, try to recognize every possible word that the users generate, and use empirical data to determine what they mean by that word.

We inject a cautionary note here: when word meanings are to be surmised from word usage data, there is always a risk that the same word may have been used in reference to several objects in the system.

In such a case, the system would not have a unique guess. (This was actually an implicit problem discussed for several of the earlier models.) In our simulations, we limited the system to returning just its single best (Model 5, Section 4.5) or best three (Model 6, Section 4.6) guesses as to what was wanted, given the word the user said. For discussion here we will mention only this last model (6c). The results in Fig. 4 show what can be achieved if all words are saved by the system, and it uses behavioral data to make the three best guesses as to what was meant. (Here too we can only present estimates of upper and lower bounds on performance.)

These results are encouraging. The big improvement is due primarily to the fact that the earlier methods, like most in current use, failed to recognize all the rare words. The unfortunate truth is that the majority of words used are "rare" words, and so it is a mistake not to reckon with them. Indeed, it turns out (see Section 4.3.4.3) that less common words tend to be more precise, more discriminating, and so are particularly valuable for the computer to know.

While we studied quite a wide variety of intermediate models, this last one really is the best. At the risk of belaboring the point, here it is summarized again. It requires: (1) the computer to keep all words that people use; (2) that data be collected on what users use each word to refer to; (3) that when someone uses a word, the computer goes to the data table and looks up the top few candidates to determine what its interpretation should be (what the word is most probably referring to); and (4) the computer presents the candidates to the user as a choice, since the interpretations may be in error. This model amounts, really, to a rich, empirically (i.e., behaviorally) defined cross index. Without it, performance is near abysmal, with it, potentially quite good.

Further improvements are shown to require the input of multiple words. This would entail means for representing and evaluating relations as well as reference, and matters of logic, syntax expression, and language comprehension. Detailed analyses of the complicated usability issues that would be involved are beyond the scope of this paper, but a number of related issues were addressed in Section 4.7.

V. DISCUSSION

We have seen that the object referent of a word cannot be predicted with great accuracy from knowledge of its past referential use. The use of statistical data on reference behavior can improve the choice of a single name by roughly a factor of two over the common procedure, in which only a single designer-chosen word is stored as an entry point for each object. Even the best system input/output function of the general kind we have described, while adding another factor of two or

so to performance, can be expected to perform at well below perfect reliability. In this section, we touch on some of the reasons for this limitation and suggest some avenues for further exploration of better means for understanding object reference.

Obviously, one of the main difficulties in predicting an intended object from a provided word is synonymy. There are many different words that can refer to the same object. Even though the receiver may know several of them, the communicant (or user) may choose another. It is the long-tailed distribution of the word usage for objects, as illustrated in Fig. 1, that is the villain. Unfortunately, however, this is only a part of the problem. Indeed, our upper bound estimates assume that this problem does not exist, that we know every word that will be used, and with what probability each will be applied to every object. Still our best upper bound predictions are quite error prone. Another part of the problem is polysemy; each word means many different things and can refer to many different objects. In our observed data, words that were frequently used tended to be applied to several objects. Clearly, if a word is used for two or more objects, there is no way that we can guess from the use of that word which unique object is its referent. The more similar the objects in the domain, the more likely it is that a single word will include more than one in its scope (Section 4.5.4.2). In general, then, one might expect that the more similar the objects in a set seem to people, the more difficulty they will have in describing them uniquely and the more difficulty a receiver of their descriptions would have knowing to what they refer.

How might one interpret descriptions of objects more accurately? Observe that in the common object experiment, human subjects were able to make a single guess as to the intended object with over 80 percent success. This is well above any of our performance estimates for systems based on the statistical information in single-word input/output tables. How do people do this? What other sources of information, either in the input description or in the receiver's mind, are brought to bear? What is needed to build a system that would approach human capacities?

One possibility is that the limits we observed may be due to utilization of only a single word or phrase from the input. Perhaps if multiple words, or the combined meaning implied by conjunctions of several words and their syntactic order, were utilized, much better reference could be achieved. The conjunctive use of words primarily serves to more narrowly specify the object of discourse. This presumably increases the precision of reference and would reduce the chance that we would guess an incorrect object from a provided description. But, under models that assume the comprehender can return several guesses as to the likely object, the recall provided by a full description

would not necessarily be much greater than that provided by the unrestricted meaning of its most important word. However, if users can provide many independent words, each an essentially new attempt, the recall rate could be raised substantially, as Model 2 suggests. Thus, allowing longer, more complex specifications and learning how to understand them is one promising direction to be investigated.

Recall also that using experts to generate armchair key words did not work well, at least in the recipe data where we tested it. We also note that in informal demonstrations, programmer subjects do not fare well in providing a name for a program to be matched by other programmers. Similarly, there have been a number of studies of indexer reliability in bibliographic indexing.[14–18] These come from quite favorable circumstances, in which the index terms are chosen from restricted vocabularies and the indexers are highly trained. The chances of one indexer choosing the same categories as another, even under these circumstances, are usually disappointingly low. Thus, the chance that professional indexers will agree with the first word entered by an untrained user does not seem to offer a promising route.

A possibility for improvement that seems worthy of further investigation is the study of the structure of the conceptions or mental representations of the objects in a domain to be referenced. The polysemy aspect of the problem arises because two or more objects are not linguistically separable. If we could learn how to group such objects into "super-objects," then we could potentially improve at least our ability to predict which of these "super-objects" is being sought. Similarly, informal impressions from the swap-and-sale superordinate data suggest that certain levels of superordination give rise to more consistent naming than others. If means can be found to reveal and represent strong hierarchical structure in the concepts being named, then possibly one can choose the levels or nodes in such structure that are best represented as the objects to be found in a data set. These "super-objects" also might be more amenable to automatic reference by the means we have modeled.

We have generated statistics relevant to this hypothesis, in the double treatment given to the editing terms. The five objects in the Ed5 data are in fact "super-objects", made by condensing the 25 editing operation–text unit pairs into categories involving just the editing operation. A look back at the numbers involved shows a uniform superiority in the retrieval and discrimination of "super-objects" over that of the individual constituent objects. In part this is just because there are fewer objects to be dealt with, but our hypothesis is that performance is further enhanced by the lower overall similarity at the super-object level. Suppose a system were built to capitalize on this kind of situation, that is by first empirically discovering the lowest

level of subdivision of a given domain at which natural descriptive language is adequate. Freely chosen key-word entries might well be effective for specifying objects at this level. But the whole problem of retrieval would not yet be solved. Users who wanted to access particular subordinates of these super-objects would have to be provided with some further mechanism. A hybrid approach in which early stages of specification are done by freely chosen keys and later stages by menu selection seems a promising approach.

An especially important matter, which we have so far neglected, is the prior probabilities of a user's intending various objects. Our data have been aimed at estimation of the input/output pointer functions of words to objects, and were collected with equal numbers of occasions for nomination of key words for each object. In real life, and in a real system, people seek different objects with different frequencies, perhaps with steep distribution functions resembling Zipf's law.[9] Our ability to predict what object a person has in mind by a word could be greatly improved by taking into account its prior, unconditional likelihood of being sought. Again, our informal impressions from the kinds of descriptions provided in the common object data is that human describers and recipients take great advantage of such frequency information. For example, in specifying the Empire State Building as a tall building in midtown Manhattan, the describer probably assumes that the receiver would choose, from the large set of possible objects, the one that is most likely to be the object of specification.

It will also require further work to see how such information could be incorporated in an automated access system. A start would be to consider an adaptive system that keeps track of the frequency with which objects are sought (as well as the frequency with which the particular input is satisfied by a particular output). Then object prior-probability data could be combined with input/output conditional probability data, like that we have investigated, by the use of Bayes' rule.[19] This would almost certainly yield much better predictions than our models estimate, or any that are currently achievable in available systems. Such a scheme might also take advantage of individual differences in cases where the same people will use the system repeatedly.

There are still other plausible means for circumventing the limitations our models suggest. There are other kinds of data access devices, such as menu-driven systems, query languages, or natural language understanding systems. These approaches all have something to offer, and probably can overcome some of the deficiencies of the pure key-word entry method. But each also has problems of its own. For example, menus cannot list a very large number of alternatives at once, so the desirable feature of turning the recall or production

problem into comprehension and choice is limited. (Note also that in menus the user must understand the system's terse descriptions of objects, much as the system has to understand those of the user of key words.) When there are many objects, a menu system must use a successive search method that relies on some kind of hierarchical tree or other presentation of the relations among the objects. How to do this in a way that leads to correct user choices at each level, to good overall performance, and to acceptable convenience are unsolved issues. Query languages generally require users to input well-formed relational algebra or Boolean expressions that are unlike anything seen in the data specifications provided in our password study. Such expressions require the kind of logical thought that is known to be extremely difficult for ordinary people.[20] Natural language understanding systems, as so far implemented at least, have yet to deal adequately with the lexical reference problem. They have usually glossed over the issue by restricting themselves to very limited domains and limited lexical input, for which they can store a reasonably adequate "hand-tailored" synonym list. We suspect that when such systems are developed for use in real data applications they will have to solve the synonymy and polysemy problems in some of the same ways (e.g., by the use of statistical input/output tables and prior object probabilities that we have been suggesting here. It is probably a mistake to take a too naively optimistic view of the value of "natural language" input to a computer device. For example, although our human subjects were much better than our model automatic systems at predicting the referent of a description, it is not obvious that their success was based on being able to "understand" the natural language of the input, at least if this is taken to mean successful syntactic parsing, etc. We believe that it is also possible that human success is based largely on statistical knowledge of the likelihood of objects and the likely referents of words, and that this is a matter that could be incorporated into a system without it doing highly intelligent natural language understanding.

VI. SUMMARY

The data we have collected on people describing objects have allowed us to estimate the likely performance of several mechanisms for understanding the references of words to information objects. We have found that input/output functions based on normative naming behavior of users will work much better than systems based on a single name provided by designers. We have also shown that a system that made several best guesses, and/or allowed the user to make several tries, and then returned a menu-like set of guesses to be chosen among, would substantially improve performance beyond current popular

methods. The best approach was to focus on what the user brings to the interaction, namely a great variety of words, and for each word have the system make one or more best guesses as to what the user meant.

But such a system for understanding references will still not perform nearly as well as a human receiver would. Thus, this model of the process of reference certainly does not fully capture what goes on in people's minds. Thus, we are clearly not yet ready to hazard a theory of how humans succeed as well as they do in this task, or to propose an automated method that would do as well or better. However, we believe that the evidence and analyses reviewed here lead to some promising suggestions for further exploration in both regards.

REFERENCES

1. J. M. Carroll, "Learning, using, and designing command paradigms," Human Learning: Journal of Practical Research and Application, *1* (January 1982), pp. 31–62.
2. T. K. Landauer, K. M. Galotti, and S. Hartwell, "Natural command names and initial learning: A study of text editing terms," Commun. ACM, *26*, No. 7 (July 1983).
3. S. T. Dumais and T. K. Landauer, unpublished work.
4. G. W. Furnas, unpublished work.
5. L. M. Gomez and R. Kraut, unpublished work.
6. *Our favorite recipes: Inverness Garden Club*, private printing, 1977–78.
7. C. Claiborne, *The New York Times Cookbook*, New York: Harper and Row, 1961.
8. J. Child, L. Bertholle, and S. Beck, *Mastering the Art of French cooking*, Vol. I, New York: Knopf, 1979.
9. G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Reading, MA: Addison-Wesley, 1949.
10. G. Herdan, *Type Token Mathematics: A Textbook of Mathematical Linguistics*, Mouton: S-Gravenhage, 1960.
11. D. A. Norman, "The trouble with UNIX," Datamation, *27*, No. 12 (November 1981), pp. 139–50.
12. M. E. Lesk, "Another view," Datamation, *27*, No. 12 (November 1981), p. 146.
13. G. Keppel and B. Z. Strand, "Free association responses to the primary responses and other responses selected from the Palermo-Jenkins norms," in L. Postman and G. Keppel (Eds.), *Norms of Word Association*, New York: Academic Press, 1970, pp. 177–87.
14. R. S. Hooper, "Indexer consistency tests—origin, measurements, results, and utilization," IBM Washington Systems Center, Bethesda, MD, 1965 Congress Int. Federation for Documentation, October 10–15, 1965.
15. J. Jacoby, "Methodology for indexer reliability tests," RADC-IN-62-1, Documentation, Inc., Bethesda, MD (March 1962).
16. D. J. Rodgers, "A study of intra-indexer consistency," General Electric Company, Washington, D. C., (September 1961).
17. J. F. Tinker, "Imprecision in meaning measured by inconsistency of indexing," American Documentation, *17* (April 1966), pp. 96–102.
18. J. F. Tinker, "Imprecision in indexing (Part II)," American Documentation, *19* (July 1968), pp. 322–30.
19. V. E. McGee, *Principles of statistics: Traditional and Bayesian*, Englewood Cliffs, NJ: Prentice Hall, 1971.
20. P. C. Wason and P. D. Johnson-Laird, *Psychology of structure and reasoning: Structure and content*, Cambridge, MA: Harvard University Press, 1972.

APPENDIX A

Formal Model Structures

In Table III below we present the formal structures for each of the six models studied. The constraints critical to the definition of each

Table III—Summary of constraints defining each model

Model	Description	Words Used		Objects Referenced		Total Word-Object Pairs
		Per Object	Total	Per Word	Total	
1	One name per object	**1**	$\leq R$	$\leq C$	**C**	C
2	M names per object	**M**	$\leq R$	$\leq C$	**C**	$M \times C$
3	Distinct name for each object	**1**	**C**	**1 or 0**	**C**	C
4	Distinct names, augmented with $M-1$ extra referents	$1 \leq n \leq M$	**C**	**M or 0**	**C**	$M \times C$
5	Recognize one referent for every word	$\leq C$	**R**	**1**	C	R
6	M referents for every word	$\leq C$	**R**	**M**	C	$M \times R$

Defining constraints for the models are in boldface. C = number of columns (objects), R = number of rows (words).

model are marked with asterisks. (Note that these constraints are to be met whenever possible, and for simplicity, the numbers below assume that this is always possible.)

APPENDIX B

Evaluating the Pure Random Versions of the Models

The success of any pure random model considered in this paper is given simply by the ratio of t, the number of cells included in the system mapping, to RC, the total number of cells in the matrix. To see this simply note that any structural constraints we might impose are only defined up to a permutation of the rows and columns. Thus, consider any table and its group, i.e., all its variants obtained by row and column permutations. The table that is the cell-wise total of all these tables must, for reasons of symmetry, be everywhere the same, and its grand total must be t times the number of tables in the group. The success of any individual table is given by the dot product of the user and system matrices (treating the matrices as vectors, i.e., sum the products of corresponding cells). The total success for the group is the sum of these dot products for the members of the group. But the sum of the dot product of one vector with several others is the dot product of that vector with the sum of the others, so the total success for the group is just the dot product of the user matrix and the total matrix. But since the total matrix is uniform, and we divide by the size of the group, we conclude that the average matrix then is just t/rc times the sum of user matrix. If the user matrix is in relative frequencies, the success is a probability.

APPENDIX C

Summary of Recall Probabilities

Table IV—Summary of recall probabilities

Model	Description	Version	M	Recall Probabilities Ed5 (n = 5)	Ed25 (n = 25)	CmOb (n = 50)	Swap (n = 64)	Recp (n = 188)
1	One name per object	WGT	1	.074	.106	.117	.142	.182†
		OPT	1	.151	.194	.257	.258	.312*
				.271	.322	.329	.353	.419†
				.163	.221	.279	.337	.362‡
2	Several names per object	WGT	1	.074	.106	.117	.142	.182†
			2	.144	.205	.210	.253	.332†
			3	.211	.295	.284	.337	.445†
		OPT	1	.151	.194	.257	.258	.312*
				.163	.221	.279	.337	.362‡
			2	.263	.319	.358	.358	.490*
				.281	.369	.406	.496	.564‡
			3	.365	.424	.420	.452	.578*
				.381	.492	.478	.595	.670‡
3	Distinct name for each object	WGT	1	.073	.076	.105	.124	.086*
		"OPT"	1	.140	.109	.226	.194	.105*
4	Distinct names, augmented with M extra referents	WGT	1	.067	.076	.109	.127	.084*
			2	.125	.145	.151	.172	.149*
			3	.175	.212	.181	.196	.200*
		"OPT"	1	.140	.109	.221	.208	.105*
			2	.214	.204	.277	.266	.170*
			3	.266	.287	.306	.296	.220*
5	Recognize one referent for every word	WGT	1	.413	.153	.516	.617	.128†
		OPT	1	.489	.176	.434	.351	.129*
				.620	.353	.647	.715	.276†
				.541	.258	.686	.813	.247‡
6	M referents for every word	WGT	1	.413	.153	.516	.617	.128†
			2	.656	.264	.646	.743	.207†
			3	.811	.358	.713	.795	.266†
		OPT	1	.489	.176	.434	.351	.129*
				.541	.258	.686	.813	.247‡
			2	.699	.302	.532	.427	.203*
				.761	.416	.819	.919	.373‡
			3	.830	.405	.577	.465	.259*
				.884	.531	.881	.956	.455‡

* Split halves
† One of the several repeat rate related statistics
‡ Self-prediction

AUTHORS

George W. Furnas, B.A. (Psychology), 1974, Harvard University; Ph.D. (Psychology), 1980, Stanford University; Bell Laboratories, 1980—. Mr. Furnas is a member of the Human Information Processing Research Department. He is doing research on cognitive aspects of person-computer interaction, with emphasis on access to information structures. Particular topics of recent interest include adaptive indexing, multidimensional scaling, and a "fisheye lens" viewer for large structures. Member, Psychometric Society, Classification Society, Society for Mathematical Psychology.

Thomas K. Landauer, B.S. (Anthropology), 1954, University of Colorado, Ph.D. (Social Psychology), 1960, Harvard University; Dartmouth College,

1960–1964; Stanford University, 1964–1969; Bell Laboratories, 1969—. During his first 10 years at Bell Laboratories, Mr. Landauer studied a variety of problems involving human memory. He worked on optimal scheduling of practice and study to maximize learning, did research aimed at understanding how humans retrieve information from their own memories, and developed a mathematical model of human memory that assumes it to be a content-addressable parallel access device that imposes no inherent structure on its content. For the last three years he has done research on psychological problems of human-computer communication. Fellow, AAAS; member, ACM, American Psychological Association, Psychonomic Society, Society for Cross-Cultural Research, Classification Society.

Louis M. Gomez, B.A. (Psychology), 1974, State University of New York at Stony Brook; Ph.D. (Psychology), 1979, University of California, Berkeley; Bell Laboratories, 1979—. Mr. Gomez is a member of the Human Information Processing Research Department. His current interests are in human factors of information retrieval, and individual differences in the acquisition of computer skills.

Susan T. Dumais, B.A. (Mathematics and Psychology), 1975, Bates College; Ph.D., (Psychology), 1979, Indiana University; Bell Laboratories, 1979—. Ms. Dumais is a member of the Human Information Processing Research Department. Her primary research interests are in the area of cognitive aspects of human-computer interface. She is currently involved in work on information retrieval, including menu selection, describing categories, and browsing for information. Member, AAAS, Midwestern Psychological Association, Metropolitan Chapter of the Human Factors Society.